AFRL-SR-BL-TR-98-

0085

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 31 March 1997	3. REPORT TYPE AND DATES COVERED Final (01 Jul 95 – 31 Dec 96)

4. TITLE AND SUBTITLE Multistrategy Learning for Computer Vision	5. FUNDING NUMBERS F49620-95-1-0424
6. AUTHORS Professor Bir Bhanu	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Riverside Riverside, CA 92521	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM 110 Duncan Avenue, Room B-115 Bolling Air Force Base, DC 20332-8080	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Current IU algorithms and systems lack the robustness to successfully process imagery acquired under real-world scenario. They do not provide the necessary consistency, reliability and predictability of results. Robust 3-D object recognition, in practical applications, remains one of the important but elusive goals of IU research. With the goal of achieving robustness, our research at UCR is directed towards learning parameters, feedback, contexts, features, concepts, and strategies of IU algorithms for model-based object recognition. Our multistrategy learning-based approach is to selectively apply machine learning techniques at multiple levels to achieve robust recognition performance. At each level, appropriate evaluation criteria are employed to monitor the performance and self-improvement of the system. We developed theoretically sound approaches to recognition and to learn segmentation for robust model-based recognition. We have developed two approaches based on reinforcement learning for closed-loop object recognition in a multi-level vision system. We show that in simple real scenes with varying environmental conditions and camera motion, effective low-level image analysis and feature extraction can be performed. We show the performance improvement of an IU system combined with learning over an IU system with no learning. Our initial research using outdoor video imagery and the Phoenix algorithm has demonstrated that (a) adaptive image segmentation can provide over 3Oim-provement in performance, as measured by the quality of segmentation, over non-adaptive techniques, and (b) learning from experience can be used to improve the performance over time. We have developed some novel techniques and we have some results for context reinforced ATR using learning techniques. These results have yet to be validated on a larger dataset.

19980129 064

14. SUBJECT TERMS algorithms, reliability, predictability	15. NUMBER OF PAGES
	16. PRICE CODE

DTIC QUALITY INSPECTED 3

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Multistrategy Learning for Computer Vision

Technical Report
VISLAB - Learning - 97 - 6

July 1, 1995 to December 31, 1996
Grant Number: F49620-95-1-0424

March 25, 1997

Prepared for the
Air Force Office of Scientific Research
Sponsor Code 2416
and the
Advanced Research Project Agency

by

Bir Bhanu
Principal Investigator

University of California
College of Engineering
Riverside, CA 92521-0425

Contributors:

Bir Bhanu
Jing Peng
Xin Bao
Yong-Jian Zheng
Songnian Rong

Contents

List of Figures

List of Tables

Chapter 1

Summary

1.1 Objectives

Current IU algorithms and systems lack the robustness to successfully process imagery acquired under real-world scenario. They do not provide the necessary consistency, reliability and predictability of results. Robust 3-D object recognition, in practical applications, remains one of the important but elusive goals of IU research. With the goal of achieving robustness, our research at UCR is directed towards learning parameters, feedback, contexts, features, concepts, and strategies of IU algorithms for model-based object recognition.

Our multistrategy learning-based IU approach selectively applies machine learning techniques at multiple levels to achieve robust recognition performance. At each level, appropriate evaluation criteria are employed to monitor the performance and self-improvement of the system.

1.2 Accomplishments

1.2.1 Learning Feedback and Parameters In an IU System (Chapters 2, 3 and 4)

Problem: To develop theoretically sound approaches to control feedback which are based on the results of recognition and to learn segmentation and feature extraction parameters for robust model-based recognition.

Approach: We have developed two approaches based on reinforcement learning for closed-loop object recognition in a multi-level vision system. These approaches use the team of learning automata algorithm and the delayed reinforcement learning algorithm.

The closed-loop object recognition system evaluates the performance of segmentation and feature extraction by using the recognition algorithm as part of the evaluation function. Recognition confidence is used as a reinforcement signal to the image segmentation or feature extraction processes. By using the recognition algorithm as part of the evaluation function, the system is able to develop recognition strategies automatically, and to recognize objects accurately on newly acquired images. As compared to the genetic algorithm which simply searches a set of parameters that optimize a prespecified evaluation function, here we have a recognition algorithm as part of the evaluation function.

In order to speed up the above algorithms we have developed a general approach (chapter 4) to image segmentation and object recognition that can adapt the image segmentation algorithm parameters to the changing environmental conditions. The edge-border coincidence is used for both local and global segmentation evaluation. However, since this measure is not reliable for object recognition, it is used in conjunction with model matching in a closed-loop object recognition system. Segmentation parameters are learned using a reinforcement learning algorithm that is based on a team of learning automata and uses edge-border coincidence or the results of model matching as reinforcement signals. The edge-border coincidence is used initially to select image segmentation parameters using the reinforcement learning algorithm. Subsequently, feature extraction and model matching are carried out for each connected component which passes through the size filter based on the expected size of objects of interest in the image. The control switches between learning integrated global and local segmentation based on the quality of segmentation and model matching.

Accomplishemnts: Using the Phoenix algorithm for the segmentation of color images, a clustering-based algorithm for the recognition of occluded 2-D objects and a team of learning automata algorithm, or a delayed reinforcement learning algorithm, we show that in simple real scenes with varying environmental conditions and camera motion, effective low-level image analysis and feature extraction can be performed. We show the performance improvement of an IU system combined with learning over an IU system with no learning.

Future Work: (a) Develop a complete reinforcement learning-based system for 3-D model-based object recognition with feedback among various levels. (b) Evaluate the performance of the technique in real-world applications such as automatic target recognition (ATR) or navigation. (c) Explore the delayed reinforcement learning (RL) algorithm for evaluating robustness of an object recognition system and the amount of training data that can be generated by the RL algorithm.

1.2.2 Learning To Integrate Vusual Information(Chapter 5)

Problem: To learn algorithm parameters, develop algorithms and evaluation criteria for multisensor image segmentation and recognition from images acquired under varying environmental conditions.

Aprroach: Genetic learning and other hybrid methods such as a combination of genetic algorithms and hill climbing.

Accomplishments: Our initial research using outdoor video imagery and the Phoenix algorithm has demonstrated that (a) adaptive image segmentation can provide over 30improvement in performance, as measured by the quality of segmentation, over non-adaptive techniques, and (b) learning from experience can be used to improve the performance over time. In our current work, we show that our approach scales with respect to the number of parameters and the size of the search space. Genetic learning combined with a hill-climbing technique is able to adaptively select good segmentation parameters and to generate the best result using the least number of segmentations. ¿From experiments designed to evaluate the scalability of our approach, we find that for the case of a four Phoenix parameter set whose search space size is 1 million, we search about 0.5% of the search space.

Future Work: (a) Learning the optimal parameter settings for adaptive image segmentation of multisensor imagery, (b) learning the optimal selection of image segmentation algorithms and evaluation criteria for multi-scenario, and (c) learning the optimal sensor combinations and cross-sensor validation of segmentation results.

1.2.3 Learning To Integrate Context With Clutter Models (Chapter 6)

Problem: To integrate contextual information with clutter models for target detection and recognition. Current image metrics commonly used to characterize images do not correlate well with the performance of target recognition systems.

Approach: The contextual parameters, which describe the environmental conditions for each training example, are used in a reinforcement learning paradigm to improve the clutter models and enhance target detection performance under multi-scenario situations. New Gabor transform-based features and other statistical image features are used to capture the statistical properties of natural backgrounds in visible and FLIR images. The non-incremental self-organizing map approach commonly used in an unsupervised mode is extended, by the addition of a near-miss injection algorithm, and used as an incremental supervised learning process for clutter characterization.

Accomplishments: A fast algorithm to compute the Gabor transform of a given image has been implemented. We have implemented two new Gabor transform-based feature groups and tested their classification performance on natural backgrounds. Experimental results show that the two feature groups could capture certain characteristics of the backgrounds, which are consistent with our theoretical expectations based on the physical meaning of each attribute within the feature group.

Using 40 second generation FLIR images, four contextual parameters (time of the day, depression angle, range to the target and air temperature) and 5 feature groups, we find 10010classifying a feature cell (rectangular regions in an image) as a clutter or a target.

Future Work: (a) Prove the convergence of the stochastic reinforcement learning algorithm for multi-feature cases. (b) Test the approach on a larger data set with a variety of contextual parameters. (c) Find the most influential environmental parameters for a given sensor, find how a feature group is affected by a given environmental parameter and find if we can make a feature invariant with respect to a given environmental parameter through normalization of the sensor data.

1.2.4 Input Adaptation (Chapter 7)

Problem: To improve the performance of an IU algorithm by adapting its input data to the desired form so that it is optimal for the given algorithm.

Approach: Two general methodologies for the performance improvement of an IU system are based on optimization of algorithm parameters and adaptation of the input. Unlike the genetic learning case for adaptive image segmentation, here we focus on the second methodology and use modified Hebbian learning rules to build adaptive feature extractors which transform the input data into the desired form for a given algorithm. Learning rules are based on different loss functions and are suitable for extracting expressive or discriminating features from the input.

Accomplishments: The feasibility of the approach is shown by designing an input adaptor for a thresholding algorithm for target detection using SAR and FLIR images. The results are excellent with input adaptor compared to the case with no input adaptor.

Future Work: (a) Develop transformations from input data to salient features needed for various classes of algorithms. (b) Compare performance with/without input adaptor for algorithms used in applications such as automatic target recognition and navigation.

1.2.5 Learning Recognition Strategies(Chapter 8)

Problem: To automate acquisition of recognition strategies in dynamic environments.

Approach: Most current model-based approaches to object recognition utilize geometric descriptions of object models, i.e., they emphasize the recognition problem as a characteristic of individual object models only. Various other factors, however, may influence the outcome of recognition in a real application such as photointerpretation. These factors include contextual information, sensor type, target type, scene models, and other non-image information. Using Case-Based Reasoning (CBR), successful recognition strategies (contextual information, algorithms, features, parameters, etc.) are stored in memory as cases and are used to solve new problems.

Since there are no algorithms that show acceptable performance over all different image sets that can be input to a system, we categorize images into classes and find the best algorithm for each class. When new image is provided to recognize an object such as a particular aircraft type, the new image is first categorized into the most similar class and then processed using the best algorithm known beforehand.

Categorization of images is, however, a very difficult problem. Instead of categorizing an image, a region of interest (ROI) is classified. For training images, ROIs are acquired and divided into classes by a human operator. The best algorithm is also selected by a human operator during training. Once images are categorized, characteristics of image sets are compiled statistically. These compiled probability distributions of values for each characteristic feature are utilized to find the most similar class. Characteristic features fall into two categories: contextual information and pure image metrics information. Weather, time of image acquisition, and viewing angles are proposed as contextual information. Homogeneity factor, convexity factor, and agglomeration factor are suggested as pure image metrics information.

Accomplishments: We have developed the basic elements of the CBR paradigm. We have experimented extensively with a C-based algorithms for aircraft recognition in aerial photographs. We have written code for characterizing image data sets.

Future Work: (a) Develop a prototype system which will have all the basic elements of CBR. (b) Select the best image metrics based on the discriminating power for categorizing images. (c) Develop reasoning, adaptation and indexing approaches that will make CBR an effective approach for IU applications.

1.2.6 Learning Composite Visual Concepts(Chapter 9)

Problem: Current grouping techniques use only perceptually motivated, low-order geometrical relationships but no object model information, to assemble simple features of the same type. As a result, the full potential of grouping for solving the indexing problem has not been realized.

Approach: Discover groups that have both a simple description and are distinctive for indexing into the model database, using a variant of explanation-based learning. The use of a two-stage grouping strategy combines domain-independent perceptual grouping and model-based grouping with a database of high-order structural arrangements.

Accomplishments: We have specified the goals, prerequisites, and preliminary formalism for "inventing" significant structural groupings from multi-class primitives.

Future Work: Implement the approach and evaluate its effectiveness for grouping in various task domains.

1.3 Publications(1 October 95 and 31 December 96)

1.3.1 Published

1. B. Bhanu, S. Lee and J. Ming, "Adaptive Image Segmentation Using a Genetic Algorithm," IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, No. 12, pp. 1543-1567, December 1995.

2. B. Bhanu, S. Lee and S. Das, "Adaptive Image Segmentation Using Genetic and Hybrid Search Methods," IEEE Transactions on Aerospace and Electronic Systems, Vol. 31, No. 4, pp. 1268-1291, October 1995.

3. B. Bhanu, X. Wu, and S. Lee, "Genetic Algorithms for Adaptive Image Segmentation,"Chapter 11, in "Early Visual Learning" Edited by S. Nayar and T. Poggio, pp. 269-298, Oxford University Press, 1996.

4. S. Rong and B. Bhanu, "Modeling Clutter and Context for Target Detection in Infrared Images," IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, pp. 106-113, June 16-20, 1996.

5. J. Peng and B. Bhanu, "Closed-Loop Object Recognition Using Reinforcement Learning," IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, pp. 538-543, June 16-20, 1996.

6. Y. Zheng and B. Bhanu, "Adaptive Object Detection Based on Modified Hebbian Learning," International Conference on Pattern Recognition, Vienna, Austria, August 25-30, 1996.

7. J. Peng and B. Bhanu, "Delayed Reinforcement Learning for Closed-Loop Object Recognition," Proc. International Conference on Pattern Recognition, Vienna, Austria, Oct. 26-29 August 1996.

8. S. Rong and B. Bhanu, "Reinforcement Learning for Integrating Context with Clutter

Models for Target Detection," Proc. ARPA Image Understanding Workshop, Palm Springs, CA, pp. 1389-1394, February 12-16, 1996.

9. J. Peng and B. Bhanu, "Delayed Reinforcement Learning for Closed-Loop Object Recognition," Proc. ARPA Image Understanding Workshop, Palm Springs, CA, pp. 1429-1436, February 12-16, 1996.

10. Y. Zheng and B. Bhanu, "Performance Improvement by Input adaptation Using Modified Hebbian Learning," Proc. ARPA Image Understanding Workshop, Palm Springs, CA, pp. 1381-1388, February 12-16, 1996.

11. B. Bhanu, "Image Understanding Research at UC Riverside: Robust Recognition of Objects in Real-World Scenes," Proc. ARPA Image Understanding Workshop, Palm Springs, CA, pp. 117-128, February 12-16, 1996.

1.3.2 Accepted but not yet published

1. J. Ming and B. Bhanu, "A Multistrategy Learning Approach for Target Model Recognition, Acquisition and Refinement," Int. J. on Pattern Recognition and Artificial Intelligence.

1.3.3 Submitted but not yet accepted

1. J. Peng and B. Bhanu, "Delayed Reinforcement Learning for Closed-Loop Object Recognition," IEEE Trans. on Systems, Man and Cybernetics, (Revised).

3. J. Peng and B. Bhanu, "Robust Image Segmentation Using Reinforcement Learning," IEEE Trans. on Pattern Analysis and Machine Intelligence, (Revised).

4. Y. Zheng and B. Bhanu, "Adaptive Object Detection From Multisensor Data," IEEE Trans. on Systems, Man and Cybernetics, June 1996.

5. Y. Zheng and B. Bhanu, "Adaptive Object Detection From Multisensor Data," IEEE International Conference on Multisensor Fusion and Integration of Intelligent Systems, Washington, D.C., Dec. 8-11, 1996.

6. S. Das and B. Bhanu, "Computational Vision: A Learning Perspective," Submitted to ACM Computing Surveys, (Under Revision).

1.4 Interactions/Transitions

1.4.1 Participation/presentations at meetings, conferences, seminars

Presented papers at the DARPA Image Understanding Workshop, Feb. 1996; IEEE Conf. on Computer vision and Pattern Recognition, June 1996; International Conf. on Pattern Recognition, Aug. 1996.

1. Chair, IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 1996.

2. ARPA Image Understanding Workshop, Palm Springs, CA, Feb. 12-16, 1996.

3. Program Committee, International Workshop on Structural and Syntactic Pattern Recognition (SSPR'96), Leipzig, Germany, August 20-23, 1996.

4. Program Committee, SPIE Conference on Neural Network Applications to Image Processing, San Jose, February 1996.

5. Program Committee, Second International Conference on Multisensor Fusion and Integration for Intelligent Systems, Dec. 8-11, 1996, Washington, D.C.

1.4.2 Consultative and advisory functions

Edited a special issue of IEEE Trans. on Image Processing on Automatic Target Detection and Recognition (with Ed Zelnio of WRDC; Dan Dudgeon of MIT Lincoln Lab; Azriel Rosenfeld of Univ. of Maryland, David Casasent of Carnegie Mellon University; and Irving Reed of Univ. of Southern California). (To be Published Jan. 1997)

1.4.3 Transitions

(i) Honeywell Inc. is using genetic algorithm for adapting algorithms to multiscenarios in the RSTA (Reconnaissance, Surveillance and Target Acquisition) Program from DARPA.

Genetic learning is also being used in the focus-of-attention module for image segmentation/labeling (in the MSTAR program of ARPA).

Our publications (papers, patent and book) were the first in this area.

(ii) Our research on closed-loop object recognition and context reinforced clutter characterization using reinforcement learning will be useful for target detection, training on the fly, model acquisition, and exploitation of SAR/FLIR images for adapting image understanding algorithms to sensor operating conditions and deployment environments, as well as performance characterization of image understanding systems for image exploitation in the context of battlefield awareness concept pursued by the DOD.

We expect our research to contribute to the SAIP ACTD and MSTAR programs from DARPA and WRDC.

1.5 New discoveries, inventions, or patent disclosures

We have developed some novel techniques and we have some results for context reinforced ATR using learning techniques. These results have yet to be validated on a larger dataset.

Chapter 2

Closed-Loop Object Recognition Using Reinforcement Learning

Current computer vision systems whose basic methodology is open-loop or *filter* type typically use image segmentation followed by object recognition algorithms. These systems are not robust for most real-world applications. In contrast, the system presented here achieves robust performance by using reinforcement learning to induce a mapping from input images to corresponding segmentation parameters. This is accomplished by using the confidence level of model matching as a reinforcement signal for a *team of learning automata* to search for segmentation parameters during training. The use of the recognition algorithm as part of the evaluation function for image segmentation gives rise to significant improvement of the system performance by automatic generation of recognition strategies. The system is verified through experiments on sequences of indoor and outdoor color images with varying external conditions.

2.1 Introduction

Image segmentation, feature extraction and model matching are the key building blocks of a computer vision system for model-based object recognition [18, 87]. The tasks performed by these building blocks are characterized as the low (segmentation), intermediate (feature extraction) and high (model matching) levels of computer vision. The goal of image segmentation is to extract meaningful objects from an image. It is essentially a pixel-based processing. Model matching uses a representation such as shape features obtained at the intermediate level for recognition. It requires explicit shape models of the object to be recognized. There is an abstraction of image information as we move from low to high levels and the processing becomes more knowledge based or goal directed.

Although there is an abundance of proposed computer vision algorithms for object recognition, there have been few systems that achieve good performance for practical applications, for most such systems do not adapt to changing environments [10]. The main difficulties, typically associated with systems that are mostly open-loop or *filter* type, can be characterized as follows.

1. The fixed set of parameters used in various vision algorithms often leads to ungraceful degradation in performance.
2. The image segmentation, feature extraction and selection are generally carried out as preprocessing steps to object recognition algorithms for model matching. These steps totally ignore the effects of the earlier results (image segmentation, feature extraction, and model matching) on the future performance of the recognition algorithm.
3. Generally the criteria used for segmentation and feature extraction require elaborate human designs. When the conditions for which they are designed are changed slightly, these algorithms fail. Furthermore, the criteria themselves can be a subject of debate [12].
4. Object recognition is a process of making sequences of decisions, i.e., applying various image analysis algorithms. Often the usefulness of a decision or the results of an individual algorithm can *only* be determined by the final outcome (e.g. matching confidence) of the recognition process. This is also known as "vision-complete" problem [17], i.e., one cannot really assign labels to the image without the knowledge of which parts of the image correspond to what objects.

This paper presents a learning based vision framework in which the above problems can be adequately addressed. The underlying theory is that any recognition system whose decision criteria for image segmentation and feature extraction, etc. are developed autonomously from the outcome of the final recognition might transcend all these problems. A direct result of the theory is that the low and high level components of a vision system must interact to achieve robust performance under changing environmental conditions. Our system accomplishes this by incorporating a reinforcement learning mechanism to control the interactions of different levels within it. Specifically, the system takes the output of the recognition algorithm and uses it as a feedback to influence the performance of the segmentation process. As a result, the recognition performance can be significantly improved over time with this method.

One attractive feature of the approach is that it includes the matching or recognition component as part of the evaluation function for image segmentation in a systematic way. An additional strength is that the system develops its independent decision criteria (segmentation parameters) to best serve the underlying recognition task. It should be emphasized that our interest is not in a simple mixture of learning and computer vision, but rather in the principled integration of the two fields at the algorithmic level. Note that the goal here is to seek a general mapping from images to parameter settings of various algorithms based on recognition results. To our knowledge, however, no such approach exists in the computer vision field. Also there is no work in the neural network field (e.g., application of Neocognition [27]) for parameter adaptation of segmentation algorithms [12].

This work is most closely related to the work by Bhanu et al. [12, 13, 14], where they describe a system that uses genetic and hybrid algorithms for learning segmentation parameters. However, the recognition algorithm is not part of the evaluation function for segmentation in their system. The genetic or hybrid algorithms simply search for a set of parameters that optimize a prespecified evaluation function (based on global and local segmentation evaluation) that may not best serve the overall goal of robust object recognition. Furthermore, the papers assume that the location of the object in the image is known for specific photointerpretation application. In our work, we do not make such an assumption. We use explicit geometric model of an object, represented by its polygonal approximation, to recognize it in the image.

In addition, Wang and Binford [91] and Ramesh [69] have investigated statistical methods for performance evaluation and tuning free parameters of an algorithm. Wang and Binford [91] presented a theoretical analysis for edge estimation and showed how one can select the gradient threshold (tuning parameter) for edge detection. Ramesh [69] has developed a methodology for the analysis of computer vision algorithms and systems using system engineering principles. To characterize the performance of an algorithm he developed statistical models for ideal image features (such as edges, corners) and random perturbations at input/output of an algorithm. Additionally, prior distributions for image features are also obtained. Using these models and a criterion function, he can characterize the performance of a given algorithm as a function of tuning parameters and determine these parameters automatically. Our approach presented in this paper differs significantly from Ramesh's [69] approach. (a) Ramesh's approach is open loop, our approach is closed loop. In our approach recognition results determine how the segmentation parameters should be changed. (b) Ramesh is tuning the parameters of an individual algorithm - it is known that the optimization of individual components does not necessarily gives the optimal results for the system. We are working with a complete recognition system (segmentation, feature extraction and model matching components) and improving the performance of the complete system. (c) Ramesh builds elaborate statistical models (using the training data) that require complex processes of annotation and approximating the measured distributions with mathematical functions to be used later. Our learning approach does not build explicit statistical models. It uses geometrical models during model matching. (d) It is relatively easier to build statistical models for algorithms like edge and corner detection. For *complex* algorithms like *Phoenix* it is difficult to model the "perfect" algorithm behavior analytically since the performance of segmentation depends nonlinearly with the changes in parameter values and there are some heuristics used in the algorithm. Considering the above factors our approach is more general for the problem that we are trying to solve. We have developed a learning based approach presented in this paper.

Section 2.2 describes a general framework for reinforcement learning-based adaptive image segmentation. Section 2.3 describes the reinforcement learning paradigm and the particular reinforcement learning algorithm employed in our system. Section 2.4 presents the experimental results evaluating the system and section 2.5 concludes the paper. Two appendices describe the basic segmentation and model matching algorithms used to perform experiments for closed-loop object recognition using reinforcement learning.

2.2 Reinforcement Learning System for Segmentation Parameter Estimation

2.2.1 The Problem

Consider the problem of recognizing an object in an input image, assuming that the model of the object is given and that the precise location of the object in the image is unknown. The conventional method, shown in Figure 2.1, for the recognition problem is to first segment the input image, then extract and select appropriate features from the segmented image, and finally perform model matching using these features. If we assume that the matching algorithm produces a real valued output indicating the degree of success upon its completion, then it is natural to use this real valued output as feedback to influence the performance of segmentation and feature extraction so as to bring about system's earlier decisions favorable for more accurate model matching. The rest of the paper describes a reinforcement learning-based vision system to achieve just that.

2.2.2 Learning to Segment Images

Our current investigation into reinforcement learning-based vision systems is focused on the problem of learning to segment images. An important characteristic of our approach is that the segmentation process takes into account the biases of the recognition algorithm to

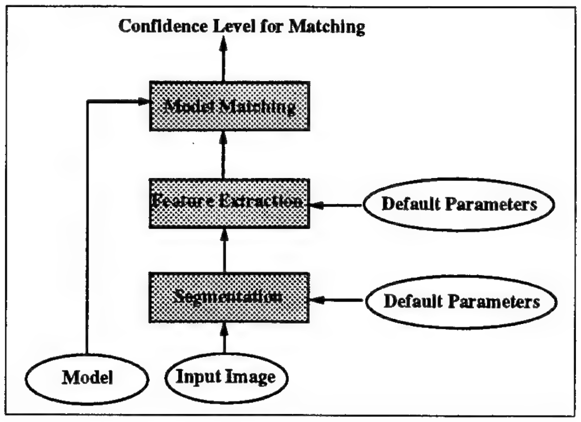


Figure 2.1: Conventional multi-level system for object recognition.

develop its own decision strategies. A consequence of this is that the effective search space of segmentation parameters can be dramatically reduced. As a result, more accurate and efficient segmentation and recognition performance can be expected.

Image Segmentation

We begin with image segmentation [40] because it is an extremely important and difficult low-level task. All subsequent interpretation tasks including object detection, feature extraction, object recognition and classification rely heavily on the quality of the segmentation process. The difficulty arises for image segmentation when only local image properties are used to define the region-of-interest for each individual object. It is known [10, 24] that correct localization may not always be possible. Thus, a good image segmentation cannot be done by grouping parts with similar image properties in a purely bottom-up fashion. Difficulties also arise when segmentation performance needs to be adapted to the changes in image quality, which is affected by variations in environmental conditions, imaging devices, lighting, etc. The following are the key characteristics [12] of the image segmentation problem: (1) When presented with a new image, selecting the appropriate set of algorithm parameters is the key to effectively segmenting the image. (2) The parameters within most segmentation algorithms typically interact in a complex, non-linear fashion, which makes it difficult to model the parameters' behavior analytically. (3) The variations between images cause changes in the segmentation results, the objective function that represents segmentation quality varies from image to image. Also, there may not be a consensus on segmentation

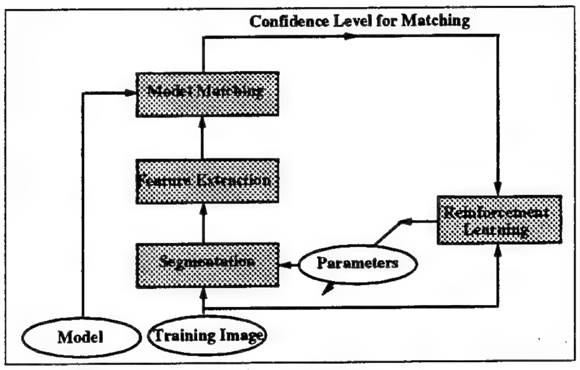


Figure 2.2: Reinforcement learning-based multi-level system for object recognition.

quality measures.

Our Approach

Each combination of segmentation parameters produces, for a given input, an unique segmentation image from which a confidence level of model matching can be computed. The simplest way to acquire high pay-off parameter combinations is through trial and error. That is, generate a combination of parameters, compute the matching confidence, generate another combination of parameters, and so on, until the confidence level has exceeded a given threshold. Better yet, if a well-defined evaluation function over the segmentation parameter space is available, then local gradient methods, such as hill-climbers, suffice. While the trial-and-error methods suffer from excessive demand for computational resources, such as time and space, the gradient methods suffer from the unrealistic requirement for an evaluation function. In contrast, reinforcement learning performs trials and errors, yet does not demand excessive computational resources; it performs hill-climbing in a statistical sense, yet does not require an evaluation function. In addition, it can generalize over unseen images as we shall see later. Furthermore, it can be easily adapted to multi-level computer vision systems. It is also feasible to construct fast, parallel devices to implement this technique for real-time applications. It thus fits our goal nicely here.

Figure 2.2 depicts the conceptual diagram of our reinforcement learning-based object recognition system that addresses the parameter selection problem encountered in image segmentation task by using the recognition algorithm itself as part of the evaluation function for image segmentation. Note that the reinforcement learning component employs a particular reinforcement learning algorithm that will be described in the next section. Figure 2.3 shows the main steps of the algorithm we use, where the algorithm terminates when either the number of iterations reaches a prespecified value (N) or the average matching confidence over entire training data (denoted by rr) has exceeded a given threshold, called R_{th}. Note that n denotes the number of images in the training set. In the event that the number of iterations has exceeded N, we will say that the object is not present in the image. Also for simplicity we assume that only one instance of the model is present in the image. Multiple instances of the model can be recognized by slight modification of the algorithm.

2.3 Reinforcement Learning

In this section we begin with a brief overview of the reinforcement learning technique. We then describe reinforcement learning algorithms applicable to our task and the modifications of these algorithms to effectively solve the problem identified in section 2.2.1.

```
• LOOP:
    1. $rr = 0$ ($rr$: average matching confidence)
    2. For each image $i$ in the training set do
        (a) Segment image $i$ using current segmentation parameters
        (b) Perform noise clean up
        (c) Get segmented regions (also called blobs or connected components)
        (d) Perform feature extraction for each blob to obtain token sets
        (e) Compute the matching of each token set against stored model and return
            the highest confidence level, $r$
        (f) $rr = rr + r$
        (g) Obtain new parameters for the segmentation algorithm using $r$ as rein-
            forcement for the reinforcement learning algorithm
• UNTIL number of iterations is equal to N or $rr/n \geq R_{th}$
```

Figure 2.3: Main Steps of the Reinforcement Learning-Based Object Recognition Algorithm.

Reinforcement learning is an important machine learning paradigm. It is a framework for learning to make sequences of decisions in an environment [6]. It is distinct from supervised learning, like the popular backpropagation algorithm, in that feedback it receives is evaluative instead of instructive. That is, for supervised learning the system is presented with the correct output for each input instance, while for reinforcement learning the system produces a response that is then evaluated using a scalar indicating the appropriateness of the response. As an example, a checker playing computer program that uses the outcome of a game to improve its performance is a reinforcement learning system. Knowledge about an outcome is useful for evaluating the total system's performance, but it says nothing about which actions were instrumental for the ultimate win or loss. In general, reinforcement learning is more widely applicable than supervised learning since any supervised learning problem can be treated as a reinforcement learning problem.

In the reinforcement learning framework, a learning system is given, at each time step, inputs describing its environment. The system then makes a decision based on these inputs, thereby causing the environment to deliver to the system a reinforcement. The value of this reinforcement depends on the environmental state, the system's decision, and possibly random disturbances. In general, reinforcement measuring the consequences of a decision can emerge at a multitude of times after a decision is made. A distinction can be made between associative and non-associative reinforcement learning. In the non-associative paradigm, reinforcement is the only information the system receives from its environment. Whereas, in the associative paradigm, the system receives input information that indicates the state of its environment as well as reinforcement. In such learning systems, a "state" is a unique representation of all previous inputs to a system. In computer vision, this state information corresponds to current input image. Our object recognition applications require us to take into account the changes appearing in the input images. The objective of the system is to select sequences of decisions to maximize the sum of future reinforcement (possibly discounted) over time. It is interesting to note that for a given state an associative reinforcement learning problem becomes a non-associative learning problem.

As noted above, a complication to reinforcement learning is the timing of reinforcement. In simple tasks, the system receives, after each decision, reinforcement indicating the goodness of that decision. Immediate reinforcement occurs commonly in function optimization problems. In more complex tasks, however, reinforcement is often temporally delayed, occurring only after the execution of a sequence of decisions. Delayed reinforcement learning is important because in many problem domains, immediate reinforcement regarding the value of a decision may not always be available. For example, in object recognition, the goodness of segmentation is not known until the recognition decision has been made. Delayed reinforcement learning is attractive and can play an important role in computer vision [66]. Because delayed reinforcement learning does not concern us here, we do not discuss this subject further.

In this paper, we instead concentrate on the immediate reinforcement learning paradigm, for it provides a simple, yet principled framework within which the main problems identified above can be properly addressed. It also serves as a stepping stone for better understanding of the issues involved in computer vision that need to be addressed by delayed reinforcement learning [66]. A well-understood method in immediate reinforcement learning is the REINFORCE algorithm [93], a class of connectionist reinforcement learning algorithms, that performs stochastic hill-climbing, and which is the subject of our paper.

2.3.1 Connectionist Reinforcement Learning

The particular class of reinforcement learning algorithms employed in our object recognition system is the connectionist REINFORCE algorithm [93], where units in such a network (depicted by the picture on the left in Figure 2.4) are *Bernoulli quasilinear units*, in that the output of such a unit is either 0 or 1, determined stochastically using the Bernoulli distribution with parameter $p = f(s)$, where f is the logistic function,

$$f(s) = 1/(1 + \exp(-s)) \tag{2.1}$$

and $s = \sum_i w_i x_i$ is the usual weighted summation of input values to that unit. For such a unit, p represents its probability of choosing 1 as its output value. The picture on the right in Figure 2.4 depicts the ith unit.

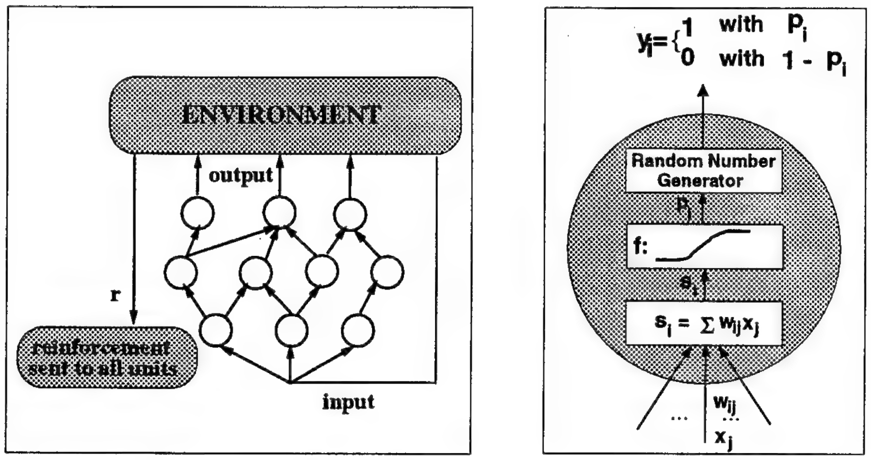

ENVIRONMENT
output
r
reinforcement sent to all units
input
$y_i = \{ \begin{smallmatrix} 1 & \text{with } p_i \\ 0 & \text{with } 1 - p_i \end{smallmatrix}$
Random Number Generator
p_i
f:
s_i
$s_i = \Sigma w_{ij} x_j$
w_{ij}
...
x_j

Figure 2.4: Left: Connectionist reinforcement learning system. Right: Bernoulli quasilinear unit.

In the general reinforcement learning paradigm, the network generates an output pattern and the environment responds by providing the reinforcement r as its evaluation of that output pattern, which is then used to drive the weight changes according to the particular reinforcement learning algorithm being used by the network. For the Bernoulli quasilinear units used in this research, the REINFORCE algorithm prescribes weight increments equal to

$$\Delta w_{ij} = \alpha(r - b)(y_i - p_i)x_j \tag{2.2}$$

where α is a positive learning rate, b serves as a *reinforcement baseline*, x_j is the input to each Bernoulli unit, y_i is the output of the ith Bernoulli unit, and p_i is an internal parameter to a Bernoulli random number generator (see equation 2.1). Note that i takes values from 1 to n and j from 1 to m, where n and m are the number of the units in the network and the number of input features, respectively.

It can be shown [93] that, regardless of how b is computed, whenever it does not depend on the immediately received reinforcement value r, and when r is sent to all the units in the network, such an algorithm satisfies

$$E\{\Delta\mathbf{W}|\mathbf{W}\} = \alpha\nabla_{\mathbf{W}}E\{r|\mathbf{W}\} \tag{2.3}$$

where E denotes the expectation operator, $\mathbf{W}$ represents the weight matrix ($n \times (m + 1)$, $m + 1$ because of m inputs plus a bias) of the network, and $\Delta\mathbf{W}$ is the change of the weight matrix. A reinforcement learning algorithm satisfying (2.3) has the property that the algorithm statistically climbs the gradient of expected reinforcement in weight space. That is, the algorithm is guaranteed to converge to a local optimum. For adapting parameters of the segmentation algorithm, it means that the segmentation parameters change in the direction along which the expected matching confidence increases. The next two subsections describe the particular network and the algorithm used in this paper.

2.3.2 The Team Architecture

We use a very simple form of trial generating network in which all of the units are output units and there are no interconnections between them. This degenerate class of network corresponds to what is called a *team* of automata in the literature on stochastic learning automata [58]. We, therefore, call these networks as *teams of Bernoulli quasilinear units.* Figure 2.5 depicts the team network used here, which corresponds directly to the reinforcement learning component in Figure 2.2. Each segmentation parameter is represented by a set of Bernoulli quasilinear units and the output of each unit is binary as we have described earlier.

For any Bernoulli quasilinear unit, the probability that it produces a 1 on any particular trial given the value of the weight matrix $\mathbf{W}$ is

$$Pr\,\{y_i = 1|\mathbf{W}\} = p_i = f(s_i) = \frac{1}{1 + e^{-s_i}}$$

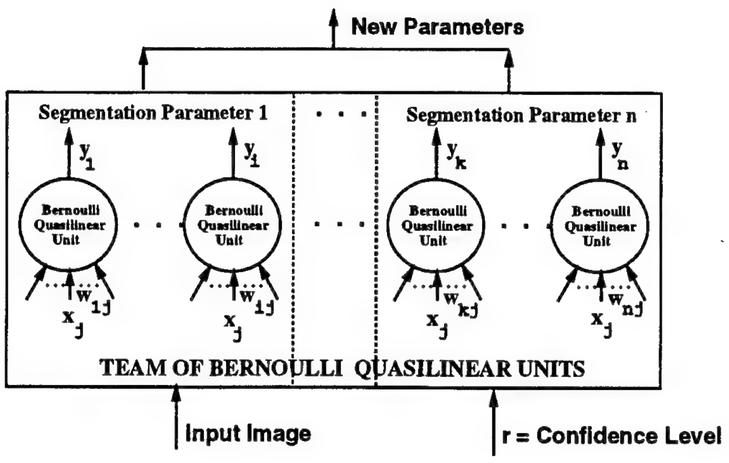


Figure 2.5: Team of Bernoulli units for learning segmentation parameters.

where $s_i = \sum_j w_{ij}x_j$. Because all units pick their outputs independently, it follows that for such a team of Bernoulli quasilinear units the probability of any particular output vector $\mathbf{y}(t)$, corresponding to an instance of segmentation parameters, conditioned on the current value of the weight matrix $\mathbf{W}$ is given by

$$Pr\{\mathbf{y}|\mathbf{W}\} = \prod_{i\in\{1,\cdots,n\}} p_i^{y_i}(1-p_i)^{1-y_i}. \tag{2.4}$$

The weights w_{ij} are adjusted according to the particular learning algorithm used. We note that when $s_i = 0$ and hence $p_i = 0.5$, the unit is equally likely to pick y_i either 0 or 1, while increasing s_i makes a 1 more likely. Adjusting the weights in a team of Bernoulli quasilinear units is thus tantamount to adjusting the probabilities (p_i's) for individual units.

Note that, except bias terms, there are no input connections in the team networks experimented in [94]. In contrast, the team network used in this paper does have input weights that play the role of long-term memory in associative learning tasks.

2.3.3 The Team Algorithm

The specific algorithm we used with the team architecture has the following form: At the t^{th} time step, after generating output $\mathbf{y}(t)$ and receiving reinforcement $r(t)$, i.e., the confidence level indicating the matching result, increment each weight w_{ij} by

$$\Delta w_{ij}(t) = \alpha(r(t) - \overline{r}(t-1))(y_i(t) - \overline{y}_i(t-1))x_j - \delta w_{ij}(t) \tag{2.5}$$

where α, the learning rate, and δ, the weight decay rate, are parameters of the algorithm. The term $(r(t) - \overline{r}(t-1))$ is called the *reinforcement factor* and $(y_i(t) - \overline{y}_i(t-1))$ the *eligibility* of the weight w_{ij} [93]. Generally, the eligibility of a weight indicates the extent to which the activity at the input of the weight was connected in the past with unit output activity. Note that this algorithm is a variant of the one described in equation (2.2), where b is replaced by $\overline{r}$ and p_i by $\overline{y}_i$.

$\overline{r}(t)$ is the exponentially weighted average, or *trace*, of prior reinforcement values

$$\overline{r}(t) = \gamma\overline{r}(t-1) + (1-\gamma)r(t) \tag{2.6}$$

with $\overline{r}(0) = 0$. The trace parameter γ was set equal to 0.9 for all the experiments reported here. Similarly $\overline{y}_i(t)$ is an average of past values of y_i computed by the same exponential weighting scheme used for $\overline{r}$. That is,

$$\overline{y}_i(t) = \gamma\overline{y}_i(t-1) + (1-\gamma)y_i(t). \tag{2.7}$$

Note that equation (2.3) does not depend on the eligibility. However, empirical study shows superior performance with this form of eligibility for function optimization [94].

The use of weight decay is chosen as a simple heuristic method to force sustained exploration of the weight space since it was found that REINFORCE algorithms without weight decay always seemed to converge prematurely. It is argued in [94] that having weight decay (the second term $\delta w_{ij}(t)$ in Equation (2.5)) is very closely related to having a nonzero mutation rate at a particular allele (feature value) in a genetic algorithm [33]. The size of the weight decay rate δ was chosen to be 0.01 in all our experiments. Note that there are other ways to force sustained exploration. One possibility is to maximize a linear combination of system's entropy and reinforcement. We omit here the detailed analysis of the method except commenting that such a strategy seeks not only a particular region of the space having high reinforcement values, but also a variety of such high value regions.

2.3.4 Implementation of the Algorithm

A different training strategy from that described in Figure 2.3 was used in the experiments reported here. Instead of looping through every image in the training set, the training procedure samples images proportional to the level of matching confidence the current system achieves. That is, the lowerer the matching confidence the system gets on an image,

```
• LOOP:
    1. For each image $i$ in the training set do
        (a) Compute matching confidence for image $i$: $CONFID_i$
        (b) $n_i = MAXCONFID - CONFID_i$
        (c) If $\sum_i n_i$ is 0, then terminate.
        (d) $proportion_i = \frac{n_i}{\sum_i n_i}$
    2. $rr = 0$ ($rr$: average matching confidence)
    3. For k = 1 to n do
        (a) Sample image $i$ according to $proportion_i$
        (b) Segment image $i$ using current segmentation parameters
        (c) Perform noise clean up
        (d) Get segmented regions (also called blobs or connected components)
        (e) Perform feature extraction for each blob to obtain token sets
        (f) Compute the matching of each token set against stored model and return
            the highest confidence level, $r$
        (g) Obtain new parameters for the segmentation algorithm using $r$ as rein-
            forcement for the team REINFORCE algorithm
        (h) $rr = rr + r$
• UNTIL number of iterations is equal to N or $rr/n \geq R_{th}$
```

Figure 2.6: Main Steps of the Proportional Training Algorithm.

the more likely the image will be sampled. In this way training is focused on those images having the lowest matching confidence, and thus faster performance improvement can be achieved. A similar technique is also adopted in [23]. Figure 2.6 shows the main steps of the proportional training algorithm, where $MAXCONFID$ (=1 in this paper) is the maximum confidence level the system can achieve, i.e., when a perfect matching occurs, n is the number of images in the training set, and N and R_{th} are input parameters to the algorithm.

2.4 Experimental Results

This section describes experimental results evaluating the performance of our system on a variety of data, including two sets of color images, one of which is indoor and the other is outdoor, and a large set of simulated data. The system has been implemented on a

SUN Ultra-1 workstation. It takes about 6 seconds to complete an iteration on a 120 by 160 image, roughly 15% of which is taken by the Phoenix algorithm. Programming optimizations can reduce the expense per iteration further.

The *Phoenix* algorithm [46] was chosen as the image segmentation component in our system because it is a well-known method for the segmentation of color images with a number of adjustable parameters. It has been the subject of several Ph.D. theses [59, 84]. *Phoenix* works by splitting regions using histogram for color features. Appendix A provides a brief overview of the algorithm. Note that any segmentation algorithm with adjustable parameters can be used in our approach.

The *Phoenix* algorithm has a total of fourteen adjustable parameters. The four most critical ones that affect the overall results of the segmentation process are used in learning. These parameters are *Hsmooth*, *Maxmin*, *Splitmin*, and *Height*. *Hsmooth* is the width of the histogram smoothing window, where smoothing is performed with a uniformly weighted moving average. *Maxmin* defines the peak-to-valley height ratio threshold. Any interval whose peak height to higher shoulder ratio is less than this threshold is merged with the neighbor on the side of the higher shoulder. *Splitmin* defines the minimum size for a region to be automatically considered for splitting. This is an absolute value, not a percentage of the image area. *Height* is the minimum acceptable peak height as a percentage of the second highest peak. The team algorithm searches for a combination of these parameters that will give rise to a segmentation from which the best recognition can be achieved. The ranges for each of these parameters are the same as those used in [12]. Table 2.1 shows sample ranges for each of these parameters. The resulting search space is about one million sample points.

Table 2.1: Sample ranges for selected *Phoenix* parameters.

Parameter	Sampling Formula	Test Range
Hsmooth: hsindex $\in [0:31]$	hsmooth=1 + 2 $*$ hsindex	1 – 63

Maxmin: mmindex $\in [0:31]$	ep=ln(100) + 0.05 $*$ mmindex
	maxmin = exp(ep) + 0.5
		100 – 471

Splitmin: smindex $\in [0:31]$	splitmin=9 + 2 $*$ smindex	9 – 71
Height: htindex $\in [0:31]$	height=1 + 2 $*$ htindex	1 – 63

Each of the *Phoenix* parameters is represented using 5 bit Gray code that has the advantage over simple binary code in that only one bit changes between representations of two consecutive numbers. One reason for using the binary representation is its usefulness as a model of certain types of distributed adaptive decision-making [93]. Another reason is that it offers a combinatorially advantageous way of approaching learning problems having a large search space. While the same task could be learned in the original parameter space, for many types of problems, including image segmentation, the binary representation can be expected to learn much faster. Since there are 4 parameters, we have a total of 20 Bernoulli quasilinear units and each parameter corresponds to the outputs of 5 units.

The feature extraction consists of finding polygon approximation tokens for each of the regions obtained after image segmentation. The polygon approximation is obtained using a split and merge technique [15] that has a fixed set of parameters.

Object recognition employs a cluster-structure matching algorithm [15] that is based on the clustering of translational and rotational transformations between the object and the model for recognizing 2-D and 3-D objects. A breif description of the algorithm is given in Appendix B. The algorithm takes as input two sets of tokens, one of which represents the stored model and the other represents the input region to be recognized. It then performs topological matching between the two token sets and computes a real number that indicates the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive the team algorithm.

It is important to note that, in the current implementation of the system, the cluster-structure matching algorithm does not have the knowledge of actual object location in the image. It simply attempts to match the stored model against the polygonal approximation of each blob in the segmented image whose size is at least 80% of the size of the model, and at the same time does not exceed it by more than 20%. The confidence level returned is the highest value ever obtained during matching.

It is worth pointing out that, during learning, the weights are updated after each presentation of an input image. This is in direct analogy to the typical weight update procedure in connectionist networks where weights are updated according to the stochastic gradient or incremental procedure instead of the total gradient rule [47]. That is, updates take place after each presentation of a single exampler without averaging over the whole training set. Both empirical and theoretical studies show that the stochastic gradient rule converges significantly faster than the total gradient rule, especially when training set contains redundant information.

Parameters (α, γ, and δ) used in reinforcement learning are determined empirically, and they are kept constant for all images. It is interesting to note that in theory the convergence of the algorithm to a local optimum does not depend on γ and δ. In practice, however, these learning parameters do affect the speed of convergence, as shown by various empirical studies conducted by several researchers [88, 93, 94], including us. Likewise, α has to be chosen sufficiently small to prevent oscillation and ensure convergence. The experimental tests performed by us showed that once the algorithm has achieved convergence many of these parameter values give rise to good segmentation performance, as verified by us visually. The initial parameter values for the *Phoenix* algorithm are chosen at random. We expect, however, that the good starting values of the segmentation parameters affect the convergence rate. Finally, as a comparison, the segmentation results with the *Phoenix* algorithm using default parameters [46] are also obtained for feature extraction and recognition on the same tasks.

2.4.1 Results on Indoor Images

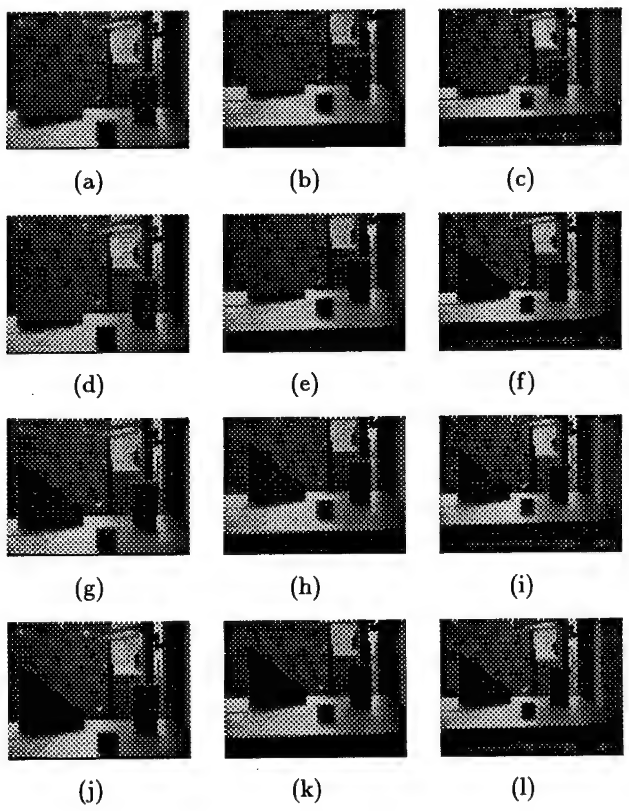


Figure 2.7: Twelve color images having simple geometric objects.

The first segmentation task whose experimental results we report here is a sequence of indoor color images (160 by 120 pixels) having simple geometric objects with varying lighting and motion conditions. These images are shown in Figure 2.7, where, from left to right, images are moving away from the camera, and within each column, lighting conditions deteriorate from top to bottom. The training set consists of the images 2.7(c), (h), (k), and (l) (randomly selected), whereas the testing data come from the rest of the images (8 images). The objective of the task is to find a set of *Phoenix*'s parameters that give rise to a segmentation of the input image that, after appropriate feature extraction, will result in the recognition of the triangular object. The model of the triangular object is represented by a polygonal approximation of its shape. The threshold for matching confidence in this case was set to 0.8. The learning rate parameter α was set to 0.008 in all the experiments. Note that, unlike previous work on image segmentation, the criteria measuring image segmentation quality here are completely determined by the matching algorithm itself.

Each unit in the team network has a total of 8 input weights. In the first experiment each of the input weights takes an average grey value of input on a 60 by 40 neighborhood on the input image plane of 120 by 160 pixels. This input image is the luminance image of the corresponding color image. Note that in this experiment the average is normalized to lie between -1 and 1. For weights that are adjacent in a unit, their receptive fields are at least 40 pixels apart in the input image. Thus, the input image is undersampled, which in turn greatly reduces the number of weights in the network. The motivation is that variations in lighting need not be adapted with high resolution.

In the second experiment each input image is projected onto the subspace spanned by the eight eigenvectors corresponding to eight largest eigen values of the original (luminance) image vector space (120 by 160 pixels). More specifically, the sample mean vector, μ, is computed as $\mu = (1/n)\sum_{i=1}^{n} \mathbf{x_i}$, where n is the number of sample vectors (in this paper n equals 12) and $\mathbf{x}$ denotes $m \times 1$ column vectors of input images. Note that here m equals 19200. A centered input matrix $\mathbf{X}$ is constructed according to

$$\mathbf{X} = (\mathbf{x}_1 - \mu, \mathbf{x}_2 - \mu, \cdots, \mathbf{x}_n - \mu).$$

Then the sample covariance matrix is obtained

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^t$$

and its eigensystem is computed, yielding eigenvalues $\lambda_i, i = 1, 2, \cdots, m$, of $\mathbf{C}$ in descending order so that $\lambda_j \geq \lambda_{j+1}$ for $j = 1, 2, \cdots, m-1$. Let $\mathbf{A}$ be a $8 \times m$ matrix whose rows are formed from the eigenvectors of $\mathbf{C}$, ordered so that the first row of $\mathbf{A}$ is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the 8th largest eigenvalue. Then new inputs are computed according to $\mathbf{z} = \mathbf{A}\mathbf{x}$ where $\mathbf{z}$ denotes 8×1 column vectors. These inputs are normalized to lie between -1 and 1. Our goal is to see which method can offer better performance. It turns out that the second method performed slightly better than the first one, as can be seen below (Figures 2.8 and 2.9). Note that, unless stated otherwise, all the figures in this section are obtained under the condition that the system takes inputs from the subspace spanned by the first 8 major axes corresponding to the eight largest eigenvalues of $\mathbf{C}$.

Figure 2.8 shows the segmentation performance (both training and testing) of the *Phoenix* algorithm with learned parameters on the images shown in Figure 2.7. The training results in Figure 2.8 are obtained after a mean value (over 5 runs) of 250 passes through the training data. Figure 2.9 shows the average confidence (over 5 runs) received by the two methods (eigen-input and mean-input) over time during training (hillclimber results are explained below under Computational Efficiency in Section 4.4). Each run consists of a sequence of trials until the average confidence level has exceeded 0.8. The threshold (0.8) serves our purpose well here since it is sufficient to demonstrate the effect of learning for object recognition.

Figure 2.10 shows the trajectory of each of the four *Hsmooth*, *Maxmin*, *Splitmin*, and *Height* parameters during training in a typical run on a particular image (in this case it is the image (c) of Figure 2.7). Note that no attempt was made to determine if the set of parameters giving rise to the final recognition is unique.

When the segmentation parameters obtained after training were applied to the images in the testing set, recognition results for all the images, but 2.7(f), are acceptable. However, if we include image 2.7(f) in the training set and allow learning to continue, experiments have been performed that show that successful recognition can be achieved for all testing images in much less time (less than 50%) compared to the time taken for training on the original training data.

In comparison, the *Phoenix* algorithm with default parameter setting was also run on the same images. Figure 2.11 shows the samples of the segmentation performance of the *Phoenix* algorithm with default parameters on the images in the first row of Figure 2.7, i.e, images 2.7((a)), 2.7(b), and 2.7 (c). These default parameters were obtained after extensive tests [46]. This default parameter setting resulted in a total matching failure.

2.4.2 Results on Outdoor Images

The second segmentation task involves a sequence of 10 outdoor color images obtained under varying environmental conditions, two of which are shown in Figures 2.12(a) and (b). These images are collected approximately every 15 minutes over approximately 2 and

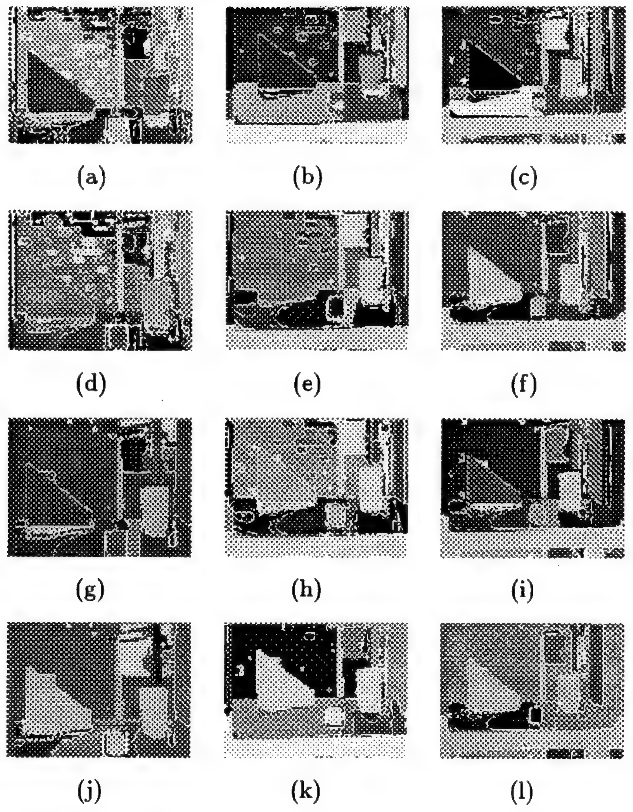


Figure 2.8: Segmentation performance of the *Phoenix* algorithm with learned parameters.

1/2 hour period [12]. The images exhibit varying shadow and reflection on the car as the position of the sun changed and clouds came in and out the field of view of the camera that had auto iris adjustment turned on. The overall goal is to recognize the car in the image. The original images are digitized at 480 by 480 pixels in size and are then subsampled to produce 120 by 120 pixel images. Five of these odd-numbered images are used as training data and five even-numbered images as testing data.

Similar to the team network for the indoor images, each unit here has a total of 9 input weights, each of which takes an average gray value of input on a 40 by 40 neighborhood on the input image plane of 120 by 120 pixels. These averages are normalized to lie between -1 and 1. Polygonal approximation of the car shown in Figure 2.12(c) is used as the model

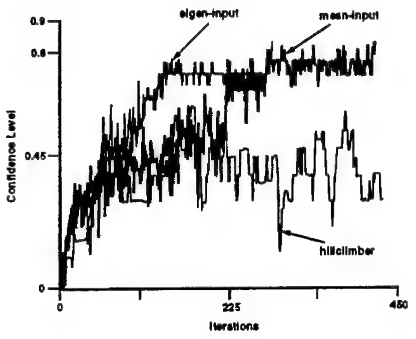


Figure 2.9: Average confidence received by the three methods over time during training.

in the cluster-structure matching algorithm. It was extracted manually in an interactive session from the first frame in the sequence.

Figure 2.13 shows a sequence of segmentations for frame 1 with *Phoenix*'s parameters sampled at iterations 20, 30, 40, 50, 60, and 74 in a particular run during training, and corresponding parameter values at each of these intervals are shown in Table 2.2. Note that

Table 2.2: Changes of parameter values during training.

Iteration	Hsmooth	Maxmin	Splitmin	Height
20	53	135	55	58
30	17	142	39	42
40	21	105	43	24
50	1	165	51	42
60	1	135	19	62
74	1	300	55	64

Figure 2.13(f) shows the final segmentation result when the highest confidence matching has been achieved. The threshold for acceptable matching confidence is set at 80% because of the low resolution of the real data.

Figures 2.14(a) and (b) show the *Phoenix* segmentation performance on two testing images (frames 2 and 4) with learned parameters obtained after training on frames 1, 3, 5, 7 and 9. For frame 2 the matching is acceptable. However, for frame 4 the result is not

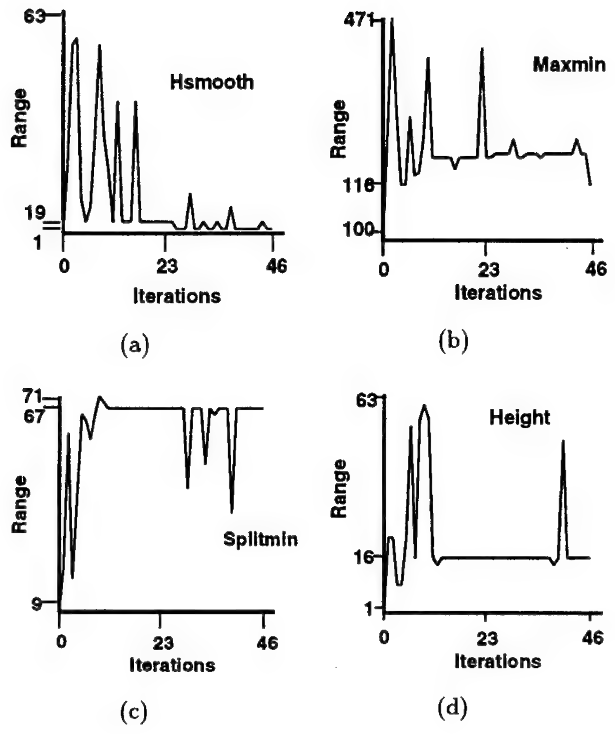


Figure 2.10: Trajectories for a particular run for each of the four parameters *Hsmooth*, *Maxmin*, *Splitmin*, and *Height* during training on a particular image (Figure 2.7(g)).

acceptable and learning is to be performed similar to the indoor examples for the adaptation of parameters.

Finally, Figures 2.14(c) and (d) show the samples of performance of *Phoenix* with default parameters on the outdoor color images shown in Figure 2.12. Note that these segmentation results are totally unacceptable.

2.4.3 Results on a Large Simulated Data Set

The simulated data experiment allows us to examine how the system will behave with a large data set. We assume the function, F, representing segmentation, feature extraction

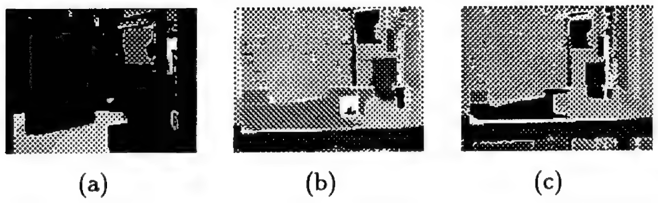


Figure 2.11: Samples of segmentation performance of the *Phoenix* algorithm with default parameters on indoor color images (Figures, 2.7(a), 2.7(b) and 2.7(c), respectively).

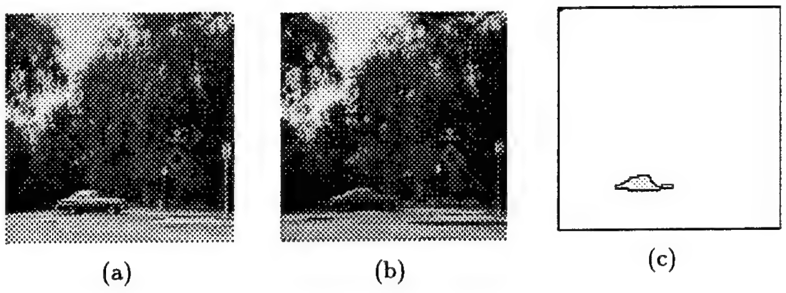


Figure 2.12: (a) and (b): Samples of outdoor color images with varying environmental conditions. (c): Polygon approximation of the car used in the matching algorithm.

and model matching components shown in Figure 2.2, is given by

$$F_{\mathbf{p}}(\mathbf{x}) = \sum_{k=0}^{3} F_{\mathbf{p}}^{k}(\mathbf{x}), \tag{2.8}$$

and

$$F_{\mathbf{p}}^{k}(\mathbf{x}) = 2.5n \prod_{i=kn/4+1}^{(k+1)n/4} (1 - |x_i - p_i|). \tag{2.9}$$

where $\mathbf{p} \in \{0,1\}^n$ is a constant. F is a mapping from the n-dimensional hypercube $\{0,1\}^n$ into the real numbers, where $n = 20$. Each point $\mathbf{x}$ in its domain is an n-dimensional bit vector.

The function $F_{\mathbf{p}}(\mathbf{x})$ is computed as follows: Divide the 20 bits into four equal-sized groups. For each group compute a score which is $2.5n$ if all the bits in that group are the

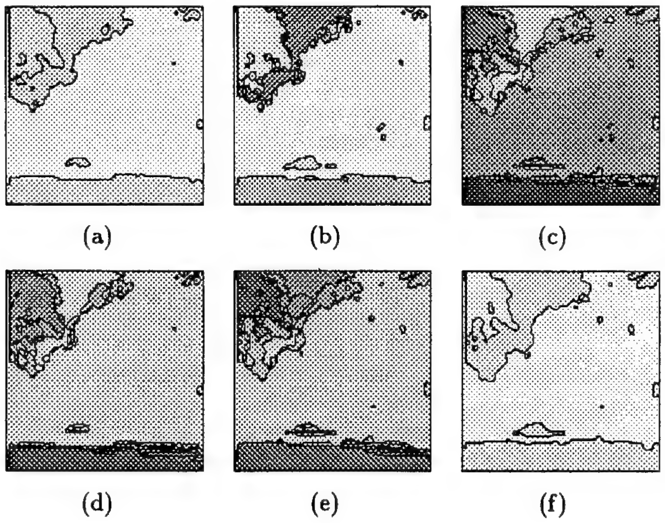


Figure 2.13: Sequence of segmentations of the first frame during training.

same as those in **p** and is 0 otherwise. Then $F_{\mathbf{p}}(\mathbf{x})$ is the sum of these four scores. This function has a global maximum of 200 at **p**. It also has very large plateaus over which the function is constant. These plateaus will confound any myopic hillclimber.

In terms of the vision system described in the paper, **x** corresponds to the encoding of segmentation parameters and $F_{\mathbf{p}}$ represents in abstract terms the matching confidence resulting from applying Phoenix with **x** to a given input image **p**.

Note that since the precise nature of the function (eq. 2.8) to be optimized is known, we can more reliably predict the strengths and limitations of the system. In this experiment, **p** is randomly generated uniformly from $\{0,1\}^n$. Then, 2000 data points whose Hamming distance to **p** is at most 4 are randomly generated from a distribution such that 80% of the data points are produced by perturbing the first 10 bits of **p**, 10% by the first 15 bits, and the rest 10% by entire 20 bits. Conceptually, each of these data points may be viewed to simulate the segmentation parameter values for an image that will give rise to the best possible recognition result for the image.

Out of these 2000 data points 500 are randomly selected as training data. The remaining 1500 data points as testing data. As in the real data experiments described above (Section 4.1), 15 normalized eigen features are computed to represent these data. Thus, there are 20 Bernoulli units, each of which has 15 input lines that encode a particular pattern to be

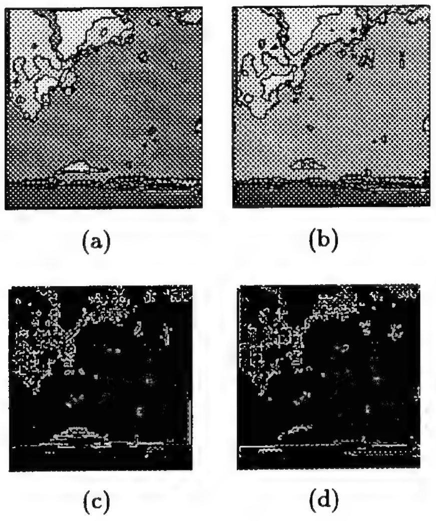


Figure 2.14: (a) and (b): Segmentation performance of the *Phoenix* algorithm on two testing images (frames 2 and 4) with learned parameters. (c) and (d): Samples of segmentation performance of the *Phoenix* algorithm with default parameters on the two images shown in Figure 2.12.

searched for.

Training consists of repeated sweeps through the training set until the average value of F has reached 190, which is about 95% of the optimal value of 200 (see eq. 2.8). An added benefit is that it prevents the system from overfitting the data, resulting in better generalization. The result shows that after about 5000 sweeps through the training data, the system achieved an average value of 180 over 90% of the testing data and an average value of 170 over the entire testing data. Further examination revealed that the majority of those testing data whose value is less than 180 come from 20 bit perturbation to $\mathbf{p}$. These data were least represented, and therefore, resulted in relatively not so good performance. This generalization characteristic is typical in connectionist networks. These results demonstrate that the algorithm can be expected to perform reasonably well on large data sets in large problem domains.

2.4.4 Computational Efficiency

The computational efficiency of the system should be evaluated against other systems having similar operating characteristics. To the best of our knowledge, however, there is no similar system that directly uses recognition result as a feedback to drive learning for image segmentation. Thus, as a comparison we applied a stochastic hillclimber to the same indoor images used for the experiments described in the above (Section 4.1). We first applied the K-Means algorithm [52] to the eigen features to determine K centers, where K = 4 in this experiment. Then four images that are closest to the four centers are used as training data. There are, therefore, four sets of *Phoenix* parameters, each of which is associated with a particular center. For a given image, generalization is made by searching for the nearest cluster center and then applying the set of *Phoenix* parameters associated with the cluster.

In the beginning, the hillclimber occasinally moves along directions that are not very promising. However, as search continues the probability of downhill movement is reduced. The annealing schedule used in this experiment is an inverse function of the number of iterations. It is improtant to note that if each dimension of the input space at every iteration has to be examined to estimate the gradient, the amount of computation required would be prohibitive. Instead, we randomly perturb 3 dimensions (where each dimension is equally likely to be selected) to move up the gradient. Thus, the amount of computation is three times of that required by the reinforcement learning system at each iteration. The decision of where to look next critically influences the computational efficiency of the optimization process. Like the reinforcement learning method, however, *a priori* gradient information is not available. It has to be estimated by sampling the search space.

A comparison of the results shown in Figure 2.9 clearly demonstrates that the reinforcement learning system performed significantly better than the stochastic hillclimber, despite the fact that it took more computation time at every iteration.

2.5 Conclusions

The key contribution of the paper is the general framework for the usage of reinforcement learning in a model-based object recognition system. Our investigation into reinforcement learning-based object recognition shows convincingly that a robust and adaptive system can be developed that automatically determines the criteria for segmentation of the input images and selects useful features that result in a system with high recognition accuracy when applied to new unseen images. Note that the performance of any learning-based computer vision system depends on the vision algorithms that are used, e.g., the recursive region-splitting *Phoenix* algorithm used in this paper for the segmentation of color images. Future research will address extensions for enlarging the scope of the approach to encompass closed-loop 3-D object recognition and problems in active vision where reinforcement learning could be extremely useful. Furthermore, incorporation of "delayed" reinforcement learning could adequately address the inherent multi-level nature of vision systems [66]. It is to be noted that in this paper we have used a parameter optimizing methodology for performance improvement. An alternate approach for performance improvement by input adaptation has been given by Zheng and Bhanu [97].

2.6 APPENDIX A: The Phoenix Segmentation Algorithm

The Phoenix image segmentation algorithm is based on a recursive region splitting technique [46]. It uses information from the histograms of the red, green, and blue image components to split regions in the image into smaller sub-regions on the basis of a peak/valley analysis of each histogram. An input image typically consists of red, green, and blue image planes, although monochrome images, texture planes, and other pixel-oriented data may also be used. Each plane is called a feature or feature plane.

Figure 2.15 shows a conceptual description of the Phoenix segmentation process. It begins with the entire image as a single region. It then fetches this region and attempts to segment it using histogram and spatial analyses. If it succeeds, the program fetches each of the new regions in turn and attempts to segment them. The process terminates when no region can be further segmented.

The histogram analysis phase computes a histogram for each feature plane, analyzes it and and selects thresholds or histogram cutpoints that are likely to identify significant homogeneous regions in the image. A set of thresholds for one feature is called an interval set. During the analysis, a histogram is first smoothed with an unweighted window average, where the window width is *hsmooth*. It is then broken into intervals such that each contains a peak and two "shoulders." A series of heuristics is applied to eliminate noise peaks. When an interval is removed, it is merged with the neighbor sharing the higher of its two shoulders. *Splitmin* is the minimum area for a region to be automatically considered for splitting.

Two tests determine if an interval should be retained. First, the ratio of peak height to the height of its higher shoulder must be greater than or equal to the *maxmin* threshold. Second, the interval area must be larger than an absolute threshold and the relative area, percent of the total histogram area. The second highest peak can now be found, and peaks lower than the *height* percent of this peak are merged. The lowest valley is then determined, and any interval whose right shoulder is higher than *absmin* (Phoenix's parameter) times this valley is merged with its right neighbor. Finally, only *intsmax* (Phoenix's parameter)

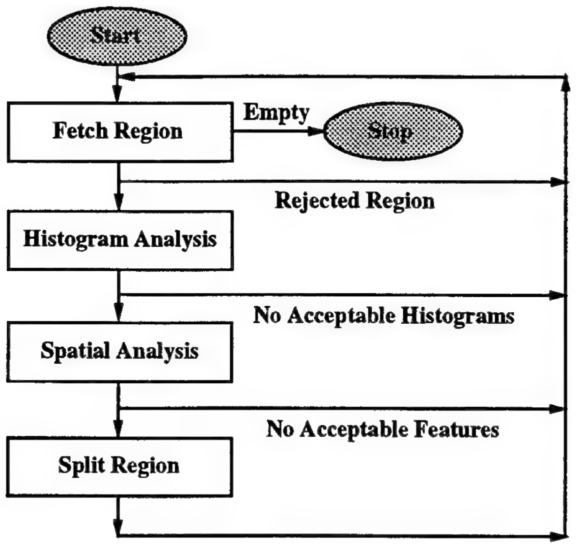


Figure 2.15: Conceptual diagram of the Phoenix segmentation algorithm.

intervals are retained by repeatedly merging intervals with low peak-to-shoulder ratio.

The spatial analysis selects the most promising interval sets, thresholds the corresponding feature planes, and extracts connected components for spatial evaluation. The feature and the interval set providing the best segmentation (the least noise area) are accepted as the segmentation feature and the thresholds.

The histogram cutpoints are now applied to the feature plane as intensity thresholds and connected components are extracted. After each feature has been evaluated, the one producing the least total noise area is accepted as the segmentation feature. If no suitable feature is found, the original region is declared terminal. Otherwise the valid patches, merged with the noise patches, are converted to new regions and added to the segmentation record. In either case, a new segmentation pass is scheduled. For additional details, see [46].

2.7 APPENDIX B: The Cluster-Structure Algorithm for Matching

The cluster-structure algorithm can be divided into the following main steps: (1) Determine Disparity Matrix,(2) Initial Clustering, (3) Sequencing, (4) Final Clustering, (5) Transform Computation. The algorithm first computes the disparity matrix. It determines the segment length of each line and the angles between successive lines from the set of vertices for the model and the image input to the program. At this point, every segment in the model will be compared against every segment in the image. If segment lengths and successor angles are compatible, the algorithm computes the rotational and translational disparity between pairs of segments. These values are stored in the disparity matrix and are indexed by the segment numbers in the model and the image. The algorithm continues until all segments have been compared. It then computes the range of rotational and translational values present in the matrix, and normalizes them over their appropriate range.

The initial clustering determines clusters from the normalized values in the disparity matrix. At each step, the program clusters all of the samples, recomputes the new cluster centers, and continues until none of the cluster centers change their positions. The program then selects the cluster having the largest number of samples. Also selected are the clusters that are within 20% of the largest one. Each cluster is considered separately and the final transform comes from the cluster that yields the highest confidence level.

The sequencing step uses the samples in the current cluster to find all sequences in the samples. This provides the critical structural information. Samples that are not placed in any sequence are discarded. The program also removes sequences that have a segment count of less than three (three segments comprise the basic local shape structure). It then computes the rotational and translation averages of each sequence that has been located.

Using the sequences and the sequence averages, the final clustering step clusters these values to find those sequences that lead to the same rotational and translational results. This is achieved by using the iterative technique of clustering, evaluating, clustering, etc. The program then selects the cluster that contains the largest number of sequences and passes this cluster to the final step.

The final step of the algorithm computes the confidence level of the transformation determined by each cluster. The cluster having the highest confidence level is selected as the final transformation cluster. It assembles the set of matched segments in the sequences in this cluster. The final output of the program is the rotation and the vertical and horizontal translation necessary to locate the model within the image. The program also produces a confidence level indicating the likelihood that the final matching is correct. For further details, see [15].

Chapter 3

Adaptive Image Segmentation and Feature Extraction

Object recognition is a multi-level process requiring a sequence of algorithms at low, intermediate and high levels. Generally, such systems are open loop with no feedback between levels and assuring their robustness is a key challenge in computer vision and pattern recognition research. A robust closed-loop system based on "delayed" reinforcement learning is introduced in this paper. The parameters of a multi-level system employed for model-based object recognition are learned. The method improves recognition results over time by using the output at the highest level as feedback for the learning system. It has been experimentally validated by learning the parameters of image segmentation and feature extraction and thereby recognizing 2-D objects. The approach systematically controls feedback in a multi-level vision system and shows promise in approaching a long-standing problem in the field of computer vision and pattern recognition.

3.1 Introduction

Most vision systems use a sequence of algorithms that operate at various stages of abstraction to perform a given task, such as object recognition. In earlier work that combines learning and vision [64], the inherent multi-stage nature of vision systems has not been addressed adequately. In this paper an approach that takes the output of the final stage and uses it as a feedback in a reinforcement learning framework to influence the performance of the lower stages of vision algorithms is presented. The overall system performance is improved over time with this method.

Figure 3.1 illustrates the typical approach for model-based object recognition, which is often unidirectional and without feedback. For those systems that do use feedback [90, 19], there is no learning from experience to improve future recognition performance, which is the subject of this correspondence. The segmentation and feature extraction modules shown in Figure 1 use default parameters that are usually obtained by the system designer by following a trial and error approach. However, the designer cannot anticipate all possible inputs to the algorithms; the content of the three-dimensional scene and the environmental conditions are not known *a priori*. The simultaneous adjustment of even a few system parameters is time-consuming and difficult and has yet to be solved satisfactorily for multi-stage systems. As a result, the approach shown in Fig. 3.1 is inadequate for real-world applications. The key to the performance improvement of a multi-stage object recognition system over time is the automatic adjustment of parameters of various algorithms used in the system.

3.2 Our Approach

If it is assumed that the model matching produces a confidence measure indicating the closeness of the selected features to the model, then it is natural to use this confidence as feedback to influence the system's performance for segmentation and feature extraction. The broad goal of such a scheme is to try to find, for any given image, a set of parameters for image segmentation and feature extraction in ways that minimize recognition errors. Applying a reinforcement learning algorithm to the parameters can be viewed as a means of doing just this when the matching confidence is used as reinforcement. Figure 3.2 shows a closed-loop reinforcement learning-based system to achieve this goal.

In contrast, it would be difficult, if not impossible, for a conventional search method to accomplish the same task. Simply, there are no well-defined evaluation functions at each of the stages for a method to search for. Furthermore, if a method uses the confidence of model-matching for evaluation, then it is not clear how the process should proceed in a systematic way. Finally, at each stage, any such method will have to delay its decision as to where to search next until the confidence of model-matching becomes available. However, this need not be the case for the approach presented in this paper. From a computational standpoint, therefore, our approach is more attractive since the computation can be distributed over time more evenly, which will reduce overall demands on the memory and speed.

The original contribution of this work is to provide an incremental method based on "delayed" reinforcement learning for inducing a general mapping from images to parameter settings in a multi-stage model-based object recognition system. A theoretical model is provided and its efficacy is validated using real-world data.

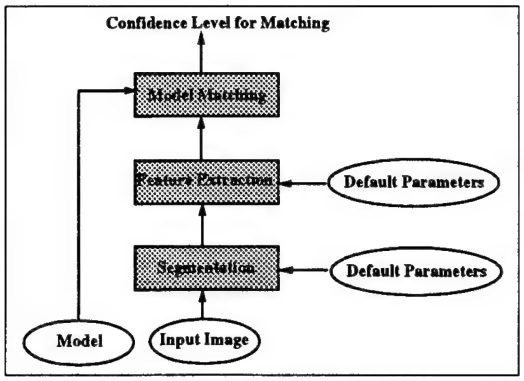


Figure 3.1: Conventional system for object recognition.

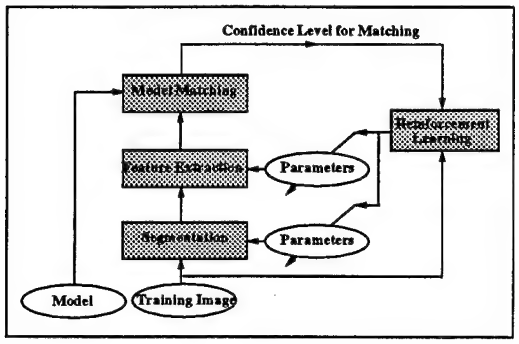


Figure 3.2: Reinforcement learning-based multi-stage system for object recognition.

3.3 Reinforcement Learning

Reinforcement learning studies computational approaches to learning from rewards and punishments (called reinforcement). It is about learning optimal control in Markov decision problems. In this paper, reinforcement corresponds to the confidence measure generated by the model matching (see Fig. 3.2). Several factors complicate reinforcement learning, the most important of which is the timing of reinforcement. Reinforcement becomes available after each action in simple tasks. In most complex tasks, however, reinforcement is often temporally delayed. For example, in the object recognition system, the goodness of segmentation and feature extraction may not be reliably known until model matching has been performed.

One set of effective methods for delayed reinforcement learning is given by the theory of dynamic programming. Given a Markov decision problem, these methods involve first determining the "optimal action-value function," the Q function [92], that assigns to each state-action pair a value measuring the average total (discounted) reward obtained when a particular action is taken in the given state and the optimal policy is followed thereafter. That is, using the notation that x denotes the current state, a the current action, r the resulting immediate reward, and y the resulting next state from taking a in x, then

$$Q(x,a) = R(x,a) + \gamma \sum_{y} P_{xy}(a) V(y) \tag{3.1}$$

where $R(x,a) = E\{r|x,a\}$ with E denoting the expectation operator, $V(x) = \max_a Q(x,a)$, $P_{xy}(a)$ is the probability of making a state transition from x to y as a result of applying action a, and $\gamma \in [0,1)$ is a discount factor. Note that once the Q function is known it is straightforward to determine the optimal policy. Note also that both x and a can be vectors.

The particular method employed in this work for learning the Q function is the Q(λ) algorithm [67], where $\lambda \in [0,1]$. Although a detailed analysis of the Q(λ) algorithm is beyond the scope of the paper, a brief explanation follows. Like Q learning [92], Q(λ) learning works by maintaining an estimate $\hat{Q}$ of the Q function and updating it so that equation (3.1) comes to be more nearly satisfied for each state-action pair encountered. In Q(λ) learning, however, the estimate $\hat{Q}(x_t, a_t)$ is regressed not just toward the estimate $\hat{V}(x_{t+1})$, which Q learning does, but to a weighted mixture of the estimates $\hat{V}(x_{t+1})$, $\hat{V}(x_{t+2})$, $\cdots$, $\hat{V}(x_{t+k})$, etc., up to and including the final outcome, where the weightings are proportional to λ^{k-1}. That is, λ controls the proportion in which future estimates are combined into overall targets. By shifting the estimate for $Q(x,a)$ toward a weighted mixture of downstream targets, Q(λ) learning not only achieves better computational efficiency, but also enables, under appropriate conditions, the elimination of the effect of initial bias. Note that it is not difficult to see that when $\lambda = 0$, Q(λ) learning reduces to simple Q learning. The typical choice for the λ value is somewhere between 0 and 1.

To implement the Q(λ) learning algorithm, a memory mechanism, called the *eligibility trace*, is used. The eligibility trace assigns a value to each experienced state-action pair with more recent ones having higher values. If $Tr(x,a)$ denotes the eligibility trace of state-action pair (x,a), then the Q(λ) learning algorithm can be described in Figure 3.3.

```
1. $\hat{Q}(x,a) = 0$ and $Tr(x,a) = 0$ for all $x$ and $a$. $t = 0$
2. Do Forever:
    (a) $x_t \leftarrow$ the current state
    (b) Choose an action $a_t$ that maximizes $\hat{Q}(x_t, a_t)$ over all $a_t$
    (c) Carry out action $a_t$ in the world. Let the short term reward be $r_t$, and the
        new state be $x_{t+1}$
    (d) $e'_t = r_t + \gamma\hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t)$ , $e_t = r_t + \gamma\hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
    (e) For each state-action pair $(x, a)$ do
        • $Tr(x,a) = \gamma\lambda Tr(x,a)$
        • $\hat{Q}_{t+1}(x,a) = \hat{Q}_t(x,a) + \alpha Tr(x,a)e_t$
    (f) $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_{t+1}(x_t, a_t) + \alpha e'_t$
    (g) $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$
    (h) $t = t + 1$
```

Figure 3.3: The Q(λ)-learning algorithm.

3.4 Reinforcement Learning for Object Recognition

In the multi-stage system for model-based object recognition described in Figure 3.2, there are unknown parameters for both the segmentation and feature extraction modules. The segmentation module is based on the *Phoenix* algorithm [46, 59]. *Phoenix* was chosen because it is a well-known method for the segmentation of color images with a number of adjustable parameters. Phoenix uses region splitting based on histograms of color features and is critically dependent on system parameters *Hsmooth* and *Maxmin*.

Hsmooth is the width of the histogram smoothing window, where smoothing is performed with a uniformly weighted moving average. *Maxmin* defines the peak-to-valley height ratio threshold. Any interval whose peak height to higher shoulder ratio is less than this threshold is merged with the neighbor on the side of the higher shoulder. The algorithm searches for a combination of these parameters that will give rise to a segmentation from which the best recognition can be achieved. The ranges for each of the two parameters are: $hsmooth = 1 + 2 * hsindex$ and $maxmin = \exp(\ln(100) + 0.05 * mmindex) + 0.5$, where $hsindex, mmindex \in [0 : 31]$. The resulting search space is about one thousand sample points.

The feature extraction module finds polygon approximation for borders of each of the regions obtained after image segmentation. It is based on a split and merge technique that is critically dependent on neighborhood parameters, called M1 and M2. They affect maximum curvature estimations [62]. For the purpose of this paper only M2 is subject to adaptation.

Our object recognition process employs a cluster-structure matching algorithm [15] that is based on the clustering of translational and rotational transformations between the object and the model for recognizing 2-D and 3-D objects. The algorithm takes as input two sets of tokens, one of which represents the stored model and the other represents the input region to be recognized. It then performs topological matching between the two token sets. It computes, based on the number of model segments that match the segments in the data, a real number that indicates the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive the algorithm.

The objective of the system is to autonomously find a set of segmentation and feature extraction parameters that achieves the maximum matching confidence for a given input image. Our model-based recognition system is a multi-stage decision process where the parameters *Hsmooth* and *Maxmin* are at the first stage of the process and the parameter M2 is at the second stage. The goodness of a particular decision such as selecting a combination of the segmentation parameters is not known until the model matching has been performed. To achieve the objective, therefore, the Q(λ) learning algorithm with the confidence of model matching as reinforcement is used to adjust the parameters at both the first stage and the second stage.

Let i be an input image to the segmentation module, $\bar{a}$ be an instance of segmentation parameters, and $\bar{b}$ be an instance of feature extraction parameters. (Note that in this paper, $\bar{b}$ is simply a scaler.) Also, let i_s be the segmented image resulting from applying *Phoenix* with $\bar{a}$ as its parameter values to image i. Then according to the Q(λ)-learning algorithm $Q(i, \bar{a})$ measures how good the instance $\bar{a}$ is when *Phoenix* applied to image i. Likewise, $Q(i_s, \bar{b})$ measures the quality of extracted features when the feature extraction algorithm with $\bar{b}$ as its parameter values is applied to the segmented image i_s. When the Q(λ) learning algorithm is applied to the parameters the value of $Q(i, \bar{a})$ will be corrected to look more like the value of the segmented image, $V(i_s) = \max_b Q(i_s, \bar{b})$, which will in turn be estimated according to the matching confidence.

```
1. Initialization: $\hat{Q}(x,\bar{p}) \leftarrow 0$ for all $x,\bar{p}$, where $x$ is either an image or a segmented
   image and $\bar{p}$ is either an instance of segmentation parameters $\bar{a}$ or feature extrac-
   tion parameters $\bar{b}$.

2. LOOP:

    • For each image $i$ in the training set do

        (a) Segment image $i$ with segmentation parameters $\bar{a} = (a_1, a_2, \cdots, a_n)$
            recommended by $\epsilon$-greedy policy; $i_s$ is the resulting segmented image.

        (b) Update $\hat{Q}(i,\bar{a})$ according to Q($\lambda$) learning with $e' = \gamma\hat{V}(i_s) - \hat{Q}(i,\bar{a})$,
            $e = \gamma\hat{V}(i_s) - \hat{V}(i)$

        (c) Perform feature extraction with feature extraction parameters $\bar{b} =$
            $(b_1, b_2, \cdots, b_n)$ recommended by $\epsilon$-greedy policy from the segmented
            image $i_s$.

        (d) Compute the matching of each connected component (which is close
            to the size of the current model) against stored model and return the
            highest confidence level $r$

        (e) Update $\hat{Q}(i_s,\bar{b})$ and $\hat{Q}(i,\bar{a})$ according to Q($\lambda$) learning with $e' =$
            $r - \hat{Q}(i_s,\bar{b})$ and $e = r - \hat{V}(i_s)$

3. UNTIL terminating condition
```

Figure 3.4: Main steps of the delayed reinforcement learning algorithm for parameter adjustment for segmentation and feature extraction.

Figure 3.4 shows the main steps of the algorithm described, where ϵ-greedy policy is a greedy policy that selects random actions for ϵ fraction of time. Although there is no strong theoretical fundation, this exploration strategy works well in practice. The algorithm terminates when either the number of iterations has exceeded a prespecified value or the recognition confidence level has reached a given threshold.

Note that in general there may be no model object, or there can be multiple instances of one model object or several different model objects in the image. If the goal is to recognize multiple objects, it might be preferable to use an average of the confidence levels resulting from each model matching. The desired result is that parameters are chosen more judiciously so as to optimize the average confidence measure that rewards parameters accommodating differences in characteristics between regions in the image. There are other possibilities as well. For example, one might design a method whose operating conditions are amended to these differences to achieve optimal performance for segmentation, feature extraction, and model matching. Such a scheme would localize its computation by ways of local segmentation to meet each individual requirement. We are currently pursuing these ideas [8]. In this paper, however, we are concerned with simple situations where only one model object is present in the image. Thus, the maximum confidence level suffices.

3.5 Experimental Validation

There are several representation schemes for the Q function in the reinforcement learning paradigm. Since the goal here is to demonstrate the effect of learning for multi-stage recognition, we have used a look-up table based representation.

The two dimensions of the look-up table are the following: (1) input or segmented (feature-extracted) image, (2) action represented by a particular combination of system parameters. The "activity" trace Tr is similarly indexed. All the table entries are initialized to zero, which means that each combination of the parameter values can be selected for evaluation with equal probability in the beginning. The focus of the experiments is to demonstrate the feasibility of using learning for multi-stage recognition. Also, $\gamma = 0.95$, $\lambda = 0.3$, and $\epsilon = 0.1$ for all the experiments reported here.

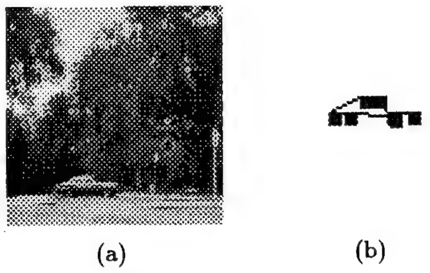


Figure 3.5: A sample outdoor color image (Frame 1 of a 20 frame sequence) and (b) polygonal model of the car.

Figure 3.5(a) shows a sample of a sequence of outdoor color images (120×120) obtained under varying environmental conditions. These images were collected approximately every 15 minutes over a $\sim$ 2 and 1/2 hour period [12]. The images exhibit varying shadow and

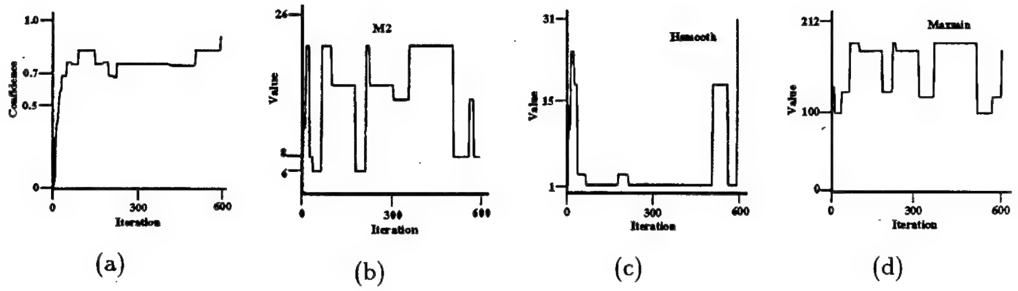


Figure 3.6: Experimental results (training) for the image in Fig. 5. (a) matching confidence level (b) parameter M2 (c) parameter HSMOOTH (d) parameter MAXMIN.

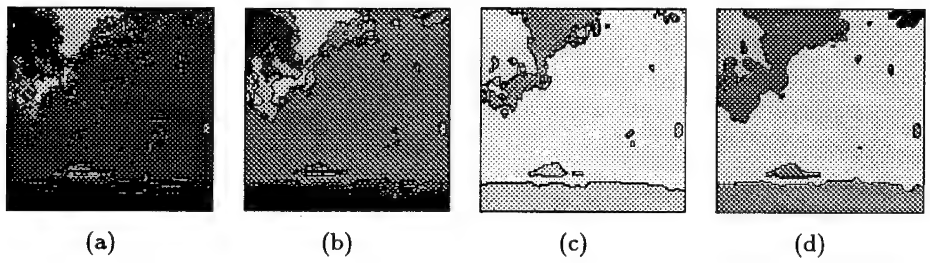


Figure 3.7: Improvement of the segmentation over time. (a) initial segmentation (b) segmentation at time step 200 (c) segmentation at time step 400 (d) segmentation at time step 600.

reflection on the car as the position of the sun changed and clouds came in and out the field of view of the camera that had auto iris adjustment turned on. The overall goal is to recognize the car in the image. It should be noted that although the image is in color, for publication purposes it is being shown in grayscale.

Figure 3.5(b) shows the 2-D model of the car located in Figure 3.5(a). The dark squares in Figure 3.5(b) correspond to labels of the vertices in the polygonal approximation of the car. The car is extracted manually in an interactive session from the first frame in the sequence and its polygonal approximation (Fig. 3.5(b)) is used as the model in the cluster-structure matching algorithm.

Figure 3.6(a) shows how the confidence, averaged over 5 runs, changes over time for the

Figure 3.8: Polygonal approximation of the car. (a) default M2 parameter (b) learned M2 parameter.

image shown in Figure 3.5(a). It should be noted that over time the confidence shown in Figure 3.6(a) increases. At the end of the training phase the confidence of the match is over 0.9 on a scale which varies between 0 and 1. For acceptable recognition, the confidence of matching has to be greater than 0.75 in the experiments reported here.

Figures 3.6(b), 3.6(c), and 3.6(d) show how the M2, *Hsmooth* and *Maxmin* change over time for a particular run, respectively. It can be seen clearly that the learned values of M2, *Hsmooth*, and *Maxmin* are considerably different from their starting random values.

To illustrate the results further, Figure 3.7 shows how the segmentation of the image improves over time during training. Figure 3.7(a) depicts the segmentation before applying the learning algorithm. Figures 3.7(b) and 3.7(c) depict the segmentation after 1/3 and 2/3 of total time (600 iterations) for training has elapsed, respectively. Figure 3.7(d) depicts the segmentation at the end of the training phase. It can be seen that the results improve considerably. While Figure 3.8(a) shows the extracted features (polygonal approximation of the car) using the default M2, parameter Figure 3.8(b) shows the same feature using the learned parameter that results in high matching confidence.

Figure 3.9(a) shows another sample frame in the image sequence in which the car of Figure 3.5(b) must be identified. It can be seen that the lighting conditions in the outdoor image is significantly different from the one shown in Fig. 3.5(a). The image is taken at a different time from Figure 3.5(a). Figure 3.9(b) shows the segmentation with default Phoenix parameters. It should be noted that when default parameters are used the car is broken up into many small blobs from which polygonal approximation of the car cannot be accurately obtained. Figures 3.9(c) and 3.9(d) show the segmentation obtained by using the parameters obtained from learning, and the polygonal approximation of the car obtained from the segmented image, respectively. The confidence of model matching is 0.88.

Figure 3.10(a) shows an indoor color image. The large triangular shaped object (wedge) is the object of interest for recognition. Figure 3.10(b) shows the segmentation result using default parameters from which the appropriate polygonal approximation of the triangular

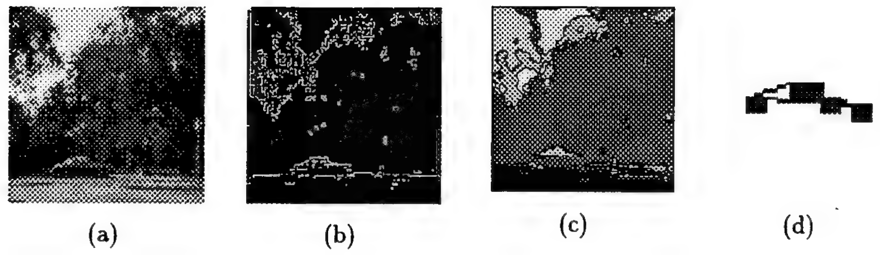


Figure 3.9: Experimental results. (a) third frame of the image sequence (b) segmentation with default parameters (c) segmentation with learned parameters (d) final polygonal approximation for the car.

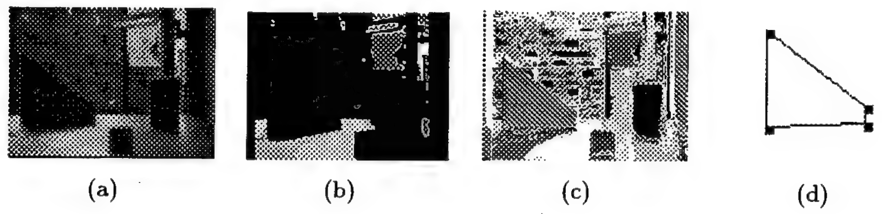


Figure 3.10: Experimental results on an indoor image. (a) the color image (b) segmentation with default parameters (c) segmentation with learned parameters (d) final polygonal approximation for the wedge.

object cannot be obtained. Figures 3.10(c) and 3.10(d) show the segmentation and the polygonal approximation of the triangular object with learned parameters respectively. In this case the confidence of model matching is 0.90. It took about a few hundred iterations to obtain the result shown in Figure 3.10(c) and (d).

3.6 Conclusion

The model-based system presented in this paper uses the recognition component as part of the evaluation functions for controlling feedback and learning parameters for image segmentation and feature extraction in a systematic way. In the experiments we have used a look-up table to represent the Q function. However, look-up table representation may not be adequate in large problems since the search space is often too large to allocate entire memory. A possible solution to this problem is to first classify input images into representative clusters [12], for example, using algorithms such as the K-Means algorithm, and then allocate memory only to the centers of these clusters. For a given image, generalization can be made by searching for the nearest cluster center. In general, however, compact function representation schemes that can generalize across spaces must be sought.

One additional benefit of the approach, due to the stochastic nature of reinforcement learning, is that the system is capable of exploring a significant portion of the search space, resulting in the discovery of good solutions, which, in general, cannot be achieved by any deterministic or simple supervised learning methods. Although we have used a three stage system to demonstrate a general approach to multi-stage model-based object recognition in a reinforcement learning paradigm, it can certainly be extended to systems having any number of stages. There is no doubt however that computational complexity will increase as the number of stages goes up.

Finally, if vision systems could be designed in one-stage as a single black box, the "simple" reinforcement paradigm would have sufficed. Earlier work on one-stage systems used a team of stochastic semi-linear units for learning image segmentation parameters [64]. In reality, however, vision systems have multiple stages for real-world tasks with parameters that need to be adjusted at each stage. Delayed reinforcement learning based approach presented here shows promise in providing a potential solution to the problem of object recognition in multi-stage systems.

Chapter 4

Integrated Image Segmentation and Object Recognition

This paper presents a general approach to image segmentation and object recognition that can adapt the image segmentation algorithm parameters to the changing environmental conditions. Segmentation parameters are learned using a reinforcement learning (RL) algorithm that is based on a team of learning automata and operates separately in a global or local manner on an image. The edge-border coincidence is used as a short term reinforcement to reduce the computational expense due to model matching during the early stage of object recognition. However, since this measure is not reliable for object recognition, it is used later in conjunction with model matching in a closed-loop object recognition system that uses the results of model matching as a reinforcement signal in a "biased" learning system. The control switches between learning integrated global and local segmentation based on the quality of segmentation and model matching. Results are presented for both indoor and outdoor color images where the performance improvement is shown for both image segmentation and object recognition with experience.

4.1 Introduction

A model based object recognition system has three key components: image segmentation, feature extraction, and model matching. The goal of image segmentation is to extract meaningful objects from an input image. Image segmentation is an important and one of the most difficult low-level computer vision tasks [13]. All subsequent tasks including feature extraction, model matching, rely heavily on the quality of the image segmentation process.

The inability to adapt the image segmentation process to real-world changes is one of the fundamental weaknesses of typical model-based object recognition systems. Despite the large number of image segmentation algorithms available [11], no general methods have been found to process the wide diversity of images encountered in real world applications. Usually, an object recognition system is *open-loop*. Segmentation and feature extraction modules use default algorithm parameters, and generally work as pre-processing steps to the model matching component. The fixed sets of algorithm parameters used in various image segmentation and feature extraction algorithms generally degrade the system performance and lack adaptability in real-world applications. These default sets of algorithm parameters are usually obtained by the system designer by following a trial and error method. Parameters obtained in this way are not robust, since when the conditions for which they are designed are changed slightly, these algorithms generally fail without any graceful degradation in performance.

The usefulness of a set of algorithm parameters in a system can only be determined by the system's output, i.e., recognition performance. To recognize different objects or instances of the same object in an image, we may need different sets of parameters locally due to the changes in local image properties, such as brightness, contrast, etc. Also the changing environmental conditions (such as the time of the day, weather conditions, etc.), affect the appearance of an image which requires the capability to adapt the representation parameters for multi-scenario object recognition. To achieve robust performance in real-world applications, a need exists to apply learning techniques which can efficiently search image segmentation and feature extraction algorithm parameter spaces and find parameter values which yield optimal results for the given recognition task. In this paper, our goal is to develop a general approach to a learning integrated model-based object recognition system, which has the ability to continuously adapt to normal environmental variations.

In the remainder of the section 1, we present an overview of the approach, related work and the contributions of the paper. Section 2 gives the details of the approach and discusses algorithms used in this research. Section 3 provides the experimental results for segmentation and recognition on both indoor and outdoor color images. Finally, section 4 presents the conclusions and the future work.

4.1.1 Overview of the Approach

In this paper, we present a general approach to reinforcement learning integrated image segmentation and object recognition. A reinforcement learning system is integrated into the model-based object recognition system to close the loop between model matching and

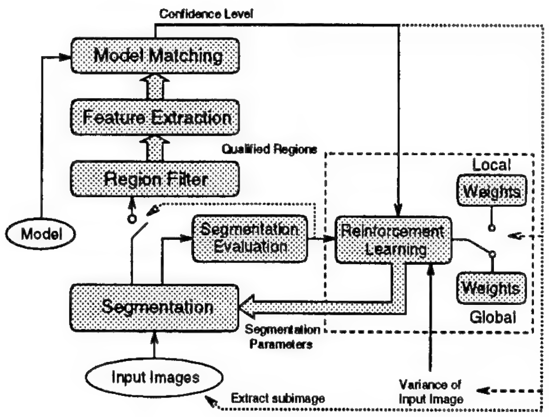


Figure 4.1: Reinforcement learning integrated image segmentation and object recognition system.

image segmentation. The basic assumption is that we know the models of the objects that are to be recognized, but we do not know the number of objects and their locations in the image. The goal of the system is to maximize the matching confidence by finding a set of image segmentation algorithm parameters for the given recognition task (We have not discussed the problem of feature extraction parameters in this paper. It is described in a separate paper by Peng and Bhanu [66]). Thus, we address the problem of adaptive segmentation as finding a set of parameters for the given model and given input image. It reflects the fact that there may not exist a single set of "optimal" parameters which can be used for recognizing different objects in a given image. Figure 4.1 provides an overview of the system. Basically, the system consists of image segmentation, feature extraction, model matching, and reinforcement learning modules. The image segmentation component extracts meaningful objects from input images, feature extraction step performs polygonal approximation of connected components, and the model matching step tells us which regions in the segmented image contain the recognized object. The model matching module indirectly evaluates the performance of the image segmentation and feature extraction processes by generating a real valued matching confidence indicating the degree of success. This real valued matching confidence is then used to drive learning for image segmentation parameters in a reinforcement learning framework.

Given the computational expense for performing model matching, our approach uses edge-border coincidence [14] as a segmentation evaluation measure to find an initial point from which to begin the search through weight space. However, since this measure is not reliable as matching confidence, we use it in conjunction with model matching in a closed-loop system to adapt segmentation parameters to current input image conditions. Subsequent feature extraction and model matching are carried out for each connected component which passes through the size filter based on the expected size of objects of interest in the image. The highest matching confidence is taken as the reinforcement signal. Learning takes place as a result of interactions between segmentation and model matching.

Significant differences in characteristics exist between an image and its subimages, so operating conditions are tuned to these differences to achieve optimal performance of segmentation and model matching. For example, to recognize two objects in an image or a single object at different locations, it is often difficult, if not impossible, to meet all requirements with one process. It is essential to localize computation to meet each individual requirement. Thus, we adopt a control that switches between global and local segmentation phases based on the quality of image segmentation and model macthing.

The reinforcement learning integrated image segmentation and object recognition system is designed to be fundamental in nature and is not dependent on any specific image segmentation algorithms or type of input images. Reinforcement learning requires only the goodness of the performance rather than the details of algorithms that produce the results. To represent segmentation parameters suitably in a reinforcement learning framework, the system only needs to know the segmentation parameters and their ranges. In our approach, a binary encoding scheme is used to represent the segmentation parameters. While the same task could be learned in the original parameter space, for many types of problems, including image segmentation, the binary representation can be expected to learn much faster [65]. In this sense, the system is independent of a particular segmentation algorithm used.

4.1.2 Related Work and Our Contributions

There is no published work on reinforcement learning integrated image segmentation and object recognition using multiple feedbacks. Bhanu and Lee [12] presented an image segmentation system which incorporates a genetic algorithm to adapt the segmentation process to changes in image characteristics caused by variable environmental conditions. In their approach, multiple segmentation quality measures are used as feedback. Some of these measures require ground-truth information which may not be always available. Peng and Bhanu [65] presented an approach in which a reinforcement learning system is used to close the loop between segmentation and recognition, and to induce a mapping from input images to corresponding segmentation parameters. Their approach is based on global image segementation which is not the best way to detect objects in an image; we need the capability of performing segmentation based on local image properties (local segmentation). Another disadvantage of their method is its time complexity which makes it problematic for practical application of computer vision.

For object recognition applications, the efficiency of the learning techniques is very important. How to add bias, a prior or domain knowledge in a reinforcement learning based system is an important topic of research in reinforcement learning [50][22][89]. For the *RATLE* system, Maclin and Shavlik [50] accept "advice" expressed in a simple programming language. This advice is compiled into "knowledge-based" connectionist Q-learning network. They show that advice-giving can speed up Q-learning when the advice is helpful (though it need not be perfectly correct). When the advice is harmful, back propagation training quickly overrides it. Dorigo and Colombetti [22] show that by using a learning technique called learning classifier system (LCS), an external trainer working within a RL framework can help a robot to achieve a goal. Thrun and Schwartz [89] have discussed methods for incorporating background knowledge into a reinforcement learning system for robot learning.

In our approach, the edge-border coincidence is used to locate an initial good point from which to begin the search through weight space for high matching confidence values. Although as a segmentation evaluation measure the edge-border coincidence is not as reliable as the matching confidence, lower edge-border coincidence values always result in poor model matching. Likewise, higher edge-border coincidence values suggest with high probability that the current set of segmentation parameters is in a close neighborhood of the optimal one. It is an inexpensive way to arrive at an initial approximation to a set of segmentation parameters that gives rise to the optimal recognition performance. The control switches between global and local segmentation processes to optimize recognition performance. To further speed-up the learning process the reinforcement learning is biased when the model matching confidence or the edge-border coincidence is used as the reinforcement signal (note that the reinforcement learning is unbiased *initially* when the edge-border coincidence is used as the reinforcement signal). We achieve better computational efficiency of the learning system and improved recognition rates compared to the system developed by Peng and Bhanu [65].

The original contributions of the reinforcement learning integrated image segmentation and object recognition system presented in this paper are:

- To achieve *robustness* for image recognition system operating in real world, model matching confidence is used as feedback to influence the image segmentation process, and thus provide an adaptive capability.

- A RL system based on a team of learning automata is applied to represent and update *both* global and local image segmentation parameters. The learning system optimizes segmentation performance on each individual image and accumulates segmentation experience over time to reduce the effort needed to optimize future unseen images.

- Edge-border coincidence, as a segmentation evaluation measure, reduces computational costs by avoiding expensive model matching, especially during earlier stages of learning.
- Learning local segmentation parameters on subimages, which may potentially contain objects, improves the performance of object recognition system.
- Explicit bias is used in the RL based system to speed up the learning process for adaptive image segmentation.

4.2 Technical Approach

The goal of our system is to maximize the model matching confidence by finding a set of image segmentation algorithm parameters for a given recognition task. To reduce the computational expense of model matching, the edge-border coincidence is first used as evaluation function to find a set of parameters from which to begin the learning. The segmentation process has two distinct phases: global and local. While global segmentation is performed for the entire image, local segmentation is carried out only for selected subimages. For a set of input images, the system takes inputs sequentially. This is similar to human visual learning process, in which the visual stimulus are presented temporally in a sequential manner. For the first input image, since the system has no accumulated experience, we initialize the system using random value of weights in the unbiased stocastic RL algorithm. For each input image thereafter, the learning process starts from the set of segmentation parameters learned based on all the previous input images. The following are the main steps of our learning algorithm:

Initial Approximation. The edge-border coincidence is used as a short term reinforcement during earlier stages of learning to drive weight changes without going through the expensive model matching process. Once the edge-border coincidence has exceeded a given threshold, the weight changes will be driven by the matching confidence, which requires more expensive computation of feature extraction and model matching.

Learning Global Segmentation. A network of biased Bernoulli units generates a set of segmentation parameters from which segmentation is performed on the entire image. The evaluation of the segmentation process is provided by the model matching confidence, which is then used to drive changes to the weights according to the reinforcement learning algorithm. We assume that we have a prior knowledge of the size of objects of interest in the images. For those connected components which pass through the size filter based on the expected size of objects of interest in the image, we perform feature extraction and model matching. The highest matching confidence is taken as the reinforcement to the learning system. If the highest matching confidence level is above a given switching threshold, we focus image segmentation and model matching on the connected component and switch to the local search process.

Learning Local Segmentation. Once a connected component has been extracted from the input image, the local search begins to find the best fit parameters for the subimage. It starts from the current estimate of weights that resulted from global learning. Similar to global learning, the matching confidence is used to update the weights estimate, until the matching confidence reaches the accepting threshold (0.8 in our experiments) or the number of iterations reaches the *MaxLocal* (in our experiments, it is set at 20). If after *MaxLocal* loops, the matching confidence is still under the accepting threshold, we switch back to the global learning process, continue the learning from where we switched to the local search process. If the matching confidence reaches the accepting threshold, the learning process for the current input image is terminated.

4.2.1 *Phoenix* Image Segmentation Algorithm

Since we are working with color imagery in our experiments, we have selected the *Phoenix* segmentation algorithm [46][59]developed at Carnegie-Mellon University and SRI International. The *Phoenix* segmentation algorithm has been widely used and tested. It works by recursively splitting regions using histogram for color features. *Phoenix* contains seventeen different control parameters, fourteen of which are adjustable. The four most critical ones that affect the overall results of the segmentation process are selected for adaptation: *Hsmooth, Maxmin, Splitmin*, and *Height*. *Hsmooth* is the width of the histogram smoothing window. *Maxmin* is the lowest acceptable peak-to-valley height ratio. *Splitmin* represents the minimum area for a region to be automatically considered for splitting. *Height* is the minimum acceptable peak height as a percentage of the second highest peak. Each parameter has 32 possible values. The resulting search space is 2^{20} sample points. Each of the *Phoenix* parameters is represented using 5 bit binary code, with each bit represented by one Bernoulli unit. To represent 4 parameters, we need a total of 20 Bernoulli units. More details about *Phoenix* are given in the report by Laws [46].

4.2.2 Segmentation Evaluation

Given that feature extraction and model matching are computationally expensive processes, it is imperative that initial approximation be made such that overall computation can be reduced. To achieve this objective, we introduce a secondary feedback signal - segmentation

Table 4.1: Ranges for selected *Phoenix* parameters.

Parameter	Sampling Formula	Range
Hsmooth: $hs \in [0:31]$	hsmooth=1 + 2 × hs	1 – 63

Maxmin: $mm \in [0:31]$	ep=ln(100)+0.05 × mm
	maxmin = exp(ep) + 0.5
		100 – 471

Splitmin: $sm \in [0:31]$	splitmin=9 + 2 × sm	9 – 71
Height: $h \in [0:31]$	height=1 + 2 * h	1 – 63

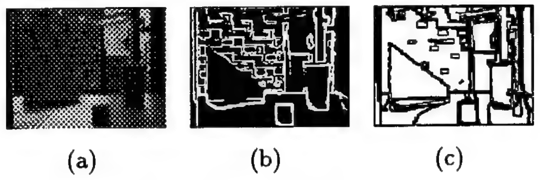


Figure 4.2: Edge-border coincidence. (a) input image; (b) Sobel edge magnitude image (threshold = 200); (c) boundaries of the segmented image. Segmentation parameters are: *Hsmooth*=7, *Maxmin*=128, *Splitmin*=47, *Height*=60.

evaluation that evaluates the image segmentation quality. There are a large number of segmentation quality measures that have been suggested. The segmentation evaluation we selected is the *edge-border coincidence* [12][53], which measures the overlap of the region borders in the segmented image relative to the edges found using an edge detector, and does not depend on any ground-truth information. In this approach, we use the *Sobel* edge detector to compute the necessary edge information. Edge-border coincidence is defined as follows. Let E be the set of pixels extracted by the edge operator and S be the set of pixels found on the region boundaries obtained from the segmentation algorithm:

$$\text{Edge} - \text{border coincidence} = \frac{n(E \cap S)}{n(E)},$$

where $n(A)$ is the number of elements in set A

Figure 4.2 shows the Sobel edge image of an experimental indoor color image and the boundaries of the segmented image using the *Phoenix* segmentation algorithm. The edge-border coincidence for the segmented image is 0.6825. Segmentation evaluation indicates the quality of the segmentation process. Matching confidence, the recognition system's output, indicates the confidence of the model matching process, and indirectly shows the segmentation quality of the recognized object. It is possible that segmentation evaluation is high and matching confidence level is low, or segmentation evaluation is low and matching confidence is high. Figure 4.3(a) shows that global segmentation evaluation is not well correlated with matching confidence. However, local segmentation evaluation, which measures the overlap between the edges and region borders of a subimage, is strongly correlated to the matching confidence, as shown in Figure 4.3(b).

Although the global segmentation evaluation does not correctly predict the matching confidence, for our purpose it is sufficient to drive initial estimates. If the edge-border coincidence is under a threshold, which indicates a low possibility to get a good recognition result, the system repeats the initial estimation process using the edge-border coincidence as the sole reinforcement feedback signal until the edge-border coincidence is greater than the threshold. At that time, the segmentation performance will be determined completely by the model matching.

4.2.3 Reinforcement Learning for Image Segmentation

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. It is appropriately thought of as a class of problems, rather than as a set of techniques [44]. This type of learning has a wide variety of applications, ranging from modeling behavior learning in experimental psychology to building active vision systems. The term *reinforcement* comes from studies of animal learning in experimental psychology. The basic idea is that if an action is followed by a satisfactory state of affairs or an improvement in the state of affairs, then the tendency to produce that action is reinforced. Reinforcement learning is similar to supervised learning in that it receives a feedback to adjust itself. However, the feedback is *evaluative* in the case of reinforcement learning. In general, reinforcement learning is more widely applicable than supervised learning and it provides a competitive approach to building autonomous learning systems that must operate in real world.

There are several reasons why we apply reinforcement learning in our computer vision system. *First*, reinforcement learning requires knowing only the goodness of the system performance rather than the details of algorithms that produce the results. In the object recognition system, model matching confidence indirectly evaluates the performance of image segmentation and feature extraction processes. It is a natural choice to select matching confidence as a reinforcement signal. *Second*, convergence is guaranteed for several reinforcement learning algorithms. *Third*, reinforcement learning is well suited to the multi-level object recognition problems in image understanding. It can systematically assign

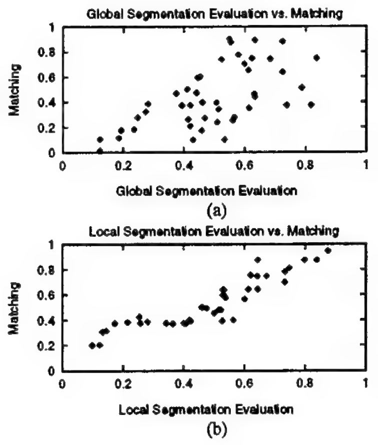


Figure 4.3: (a) Global edge-border coincidence vs. matching confidence; (b) Local edge-border coincidence vs. matching confidence for recognizing the cup in the image shown in Figure 4.2(a).

rewards to different levels in a computer vision system.

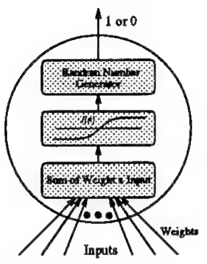


Figure 4.4: Basic structure of a Bernoulli unit.

The particular class of reinforcement learning algorithms employed in our system is the connectionist *REINFORCE* algorithm [93], where units in such a network are *Bernoulli quasi-linear units.* Figure 4.4 shows the basic structure of a Bernoulli unit. A team of five independent Bernoulli units represent a segmentation parameter with 32 possible values. The output of each unit is either 1 or 0, determined stochastically using the Bernoulli distribution with probability mass function $p = f(s)$, where f is the logistic function. For such an unit, p represents the probability of choosing 1 as its output value.

$$f(s) = \frac{1}{1+e^{-s}}, \quad \textit{where } s = \sum_j w_{ij}x_j$$

where w_{ij} is the weight of the jth input for unit i, and x_j is the jth input value for each unit. In the reinforcement learning paradigm, the learning component uses the reinforcement $r(t)$ to drive the weight changes according to a particular reinforcement learning algorithm used by the network. The specific algorithm we used has the following form: for each unit, at the tth time step, after generating output $y(t)$ and receiving reinforcement signal $r(t)$, increment each weight w_{ij} by

$$\Delta w_{ij}(t) = \alpha[r(t) - \bar{r}(t-1)][y_i(t) - \bar{y}_i(t-1)]x_j - \delta w_{ij}(t)$$

where α is the learning rate, δ is the weight decay rate, x_j is the input to each Bernoulli unit, y_i is the output of the ith Bernoulli unit. The term $r(t) - \bar{r}(t-1)$ is called the *reinforcement factor*, and $y_i(t) - \bar{y}_i(t-1)$ is the *eligibility* of the weight w_{ij}. $\bar{r}(t)$ is the exponentially weighted average of prior reinforcement values,

$$\bar{r}(t) = \gamma\bar{r}(t-1) + (1-\gamma)r(t), \quad \textit{with } \bar{r}(0) = 0$$

γ is the trace parameter. Similarly, $\bar{y}_i(t)$ is an average of past values of y_i computed by the same exponential weighted scheme used for $\bar{r}(t)$,

$$\bar{y}_i(t) = \gamma\bar{y}_i(t-1) + (1-\gamma)y_i(t)$$

The algorithm has the convergence property [93] such that it statistically climbs the gradient of expected reinforcement in weight space. The weight decay is used as a simple method to force the sustained exploration of the weight space.

Note that a team of 20 Bernoulli units represents the four image segmentation parameters selected for learning. Each bit of a parameter is independent of each other. Thus, it allows us to search the parameter space thoroughly.

4.2.4 Feature Extraction and Model Matching

Feature extraction consists of finding polygon approximation tokens for each connected component obtained after image segmentation. To speed up the learning process, we assume that we have the prior knowledge of the *approximate size* (area) of the object, and only those connected components whose area (number of pixels) are comparable with the area of the model object are approximated by a polygon. In Figure 1, the *region filter* selects

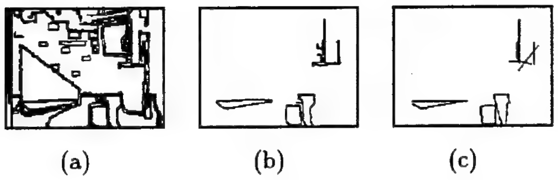


Figure 4.5: (a) Boundaries of the segmented image shown in Figure 4.2(a) (segmentation parameters are: *Hsmooth*=7, *Maxmin*=128, *Splitmin*=47, *Height*=54). (b) Selected regions whose areas are in the expected range (200 – 450 pixels), (c) Polygon approximation of these regions (parameters as specified in this section).

those connected components whose areas are in the expected range. For example, in our experiment on indoor images, the cup is the target object. The expected area is from 200 to 450 pixels. Figure 4.5 shows the boundaries of a segmented image, selected regions whose areas are in the expected range, and the polygon approximation of these regions. The polygon approximation is implemented by calling the polygon approximation routine in *Khoros* [70]. The resulting polygon approximation is a vector image to store the result of the linear approximation. The image contains two points for each estimated line. The polygon approximation has a fixed set of parameters:

- Minimal segment length for straight line - 5. When the estimated straight line has a length less than this threshold, it is skipped over.
- Elimination percentage - 0.1. Percentage of line length rejected to calculate parameters of the straight line.
- Approximation error - 0.6. Threshold Value for the approximation error. When the calculated error is greater than this value, the line is broken.

Model matching employs a cluster-structure matching algorithm [15] which is based on forming the clusters of translational and rotational transformations between the object and the model. The algorithm takes as input two sets of tokens, one of which represents the stored model and the other represents the input region to be recognized. It then performs topological matching between the two token sets and computes a real number that indicates the confidence level of the matching process. Basically, the technique consists of three steps: clustering of border segment transformations; finding continuous sequences of segments in appropriately chosen clusters; and clustering of sequence average transformation values. More details about this algorithm are given in [15].

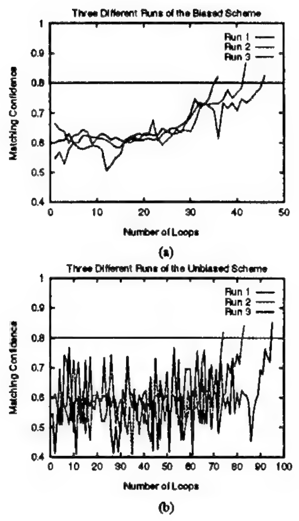


Figure 4.6: Matching confidence history of three runs of the biased and unbiased RL algorithms on the image shown in Figure 4.2(a). (a) biased; (b) unbiased.

4.2.5 Biased Reinforcement Learning for Image Segmentation

In the RL algorithm as described in section 2.3, each of the bits of each of the parameters is independent. The output of each bit depends on the value of p, which represents the probability of an unit to choose 1 as its output. In the initialization phase, we use the *unbiased* RL algorithm in which the output of each bit of a parameter is determined in the following way:

$$y_i = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

It is "unbiased" in that the output of a bit is governed solely by the Bernoulli probability law. The advantage is that rapid changes in output values allow giant leaps in the search space, which in turn enables the learning system to quickly discover suspected high pay-off regions. However, once the system has arrived at the vicinity of a local optimum, as will be the case after the initial estimation, changes in the most significant bit will drastically alter the parameter value, often jumping out of the neighborhood of the local optimum. Ideally, once the learning system discovers that it is within a possible high pay-off region, it should attempt to capture the regularities of the region. This then biases future search toward points within it. The challenge, of course, is to have a learning algorithm that allows the parameters controlling the search distribution to be adjusted so that this distribution comes to capture this knowledge. The algorithm described here shows some promise in this regard. In order to force parameters to change slowly, after the initialization phase, we apply a *biased* RL algorithm in which the two most significant bits of a parameter are forced to change in a slower fashion as:

$$y_i = \begin{cases} 1 & \text{if } p > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

and other bits use the the same rule as described in the unbiased RL algorithm. Figure 4.6 shows the experimental results of the two schemes on the image shown in Figure 4.2(a). In this experiment, we only apply the initialization followed by global learning without switching between global and local learning. The results show that the biased RL algorithm demonstrates a speed up of $2 - 3$.

4.2.6 Algorithm Description

Figure 4.7 shows the implementation of our algorithm. The algorithm works by switching between global and local segmentation. Initially, if the system has no accumulated knowledge, the edge-border coincidence is used as the evaluation function to search a set of image segmentation parameters using unbiased reinforcement learning algorithm. Otherwise, the input image is segmented using the set of parameters learned from previous images. *EB1* and *EB2* are two thresholds for edge-border coincidence. During the initial unbiased reinforcement learning phase, if the edge-border coincidence is greater than *EB1* ($= 0.5$ in our experiments), then we can start the learning process with a high expectation to generate good recognition results. During the global segmentation phase, if the segmentation quality is less than *EB2* ($= 0.4$ in our experiments), the object is less likely to be present in the segmented image, and choosing another set of parameters using the biased RL algorithm with the current reinforcement signal can speed up the process.

In the global segmentation procedure, if the global segmentation loops more than *Max-Global*, we conclude that the object does not appear in the image and terminate the learning process for the given input image. For each connected component which passes the region filter, if the matching confidence is greater than *Switch*, then we can switch the control from global to local segmentation. During local segmentation, if the matching confidence reaches *Accept*, we conclude that the connected component is the recognized model object. If the local segmentation loops more than *MaxLocal*, the control will switch back to global segmentation since the object is not likely to be extracted in the subimage and we resume the global segmentation process.

```
procedure Initialization( )
  generate a set of random weights
  repeat
    compute the segmentation parameters
    segment the image and compute edge-border coincidence
    r(i) = edge-border coincidence, update weights
  until edge-border coincidence > EB1

procedure Global_Segmentation( )
  r(0) = 0.5,  highest_matching_confidence = 0
  for i from 1 to MaxGlobal do
    for each connected component which passes the size filter do
      feature extraction and model matching
      if matching_confidence > Switch then
        Local_Segmentation( )
      if matching_confidence > highest_matching_confidence
        highest_matching_confidence = matching_confidence
    r(i) = highest_matching_confidence
    if recognized all the connected components then exit
    count = 0
    repeat
      compute the segmentation parameters using r(i)
      segment the image using the current set of parameters
      count++
      if count > MaxSeg then exit
    until edge-border coincidence > EB2

procedure Local_Segmentation( )
  extract subimage from the input image
  compute standard deviations of parts of the subimage
  copy the weights from global to local process
  count = 0
  while count < MaxLocal do
    subimage segmentation, feature extraction, and model matching
    update weights using matching confidence as reinforcement
    if matching confidence > Accept then recognized and return
    count++
```

Figure 4.7: Algorithm description.

4.3 Experimental Results

The system is verified through a set of 12 indoor and a set of 12 outdoor color images. These images are acquired at different times and different viewing distances with varying lighting conditions. The size of indoor images is 120 by 160 pixels, and the size of outdoor images is 120 by 120 pixels. Each image is decomposed into 4 images for *Phoenix* segmentation – red, green, blue components, and the Y component of YIQ model of color images. For the indoor images, the desired object is the cup in the image, and in the outdoor images, the

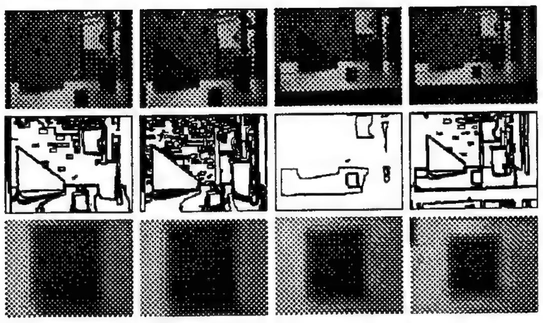


Figure 4.8: Row 1: input images; row 2, 3: corresponding segmented image and recognized object. For each input image, global segmentation evaluation, local segmentation evaluation for the selected object, and matching confidence are (0.67, 0.74, 0.87); (0.87, 0.62, 0.93); (0.22, 0.82, 0.91); (0.68, 0.73, 0.92). The learned *Phoenix* segmentation parameters *Hsmooth, Maxmin, Splitmin*, and *Height* after local learning process are (7 122 47 52); (7 128 47 52); (5 471 19 58); (11 192 59 48).

target object is the traffic sign. The expected size of the cup and the traffic sign are 200 to 450 pixels and 36 to 100 pixels, respectively.

Based on the size of the object to be recognized in the image, we divide the Y component image into 48 subimages for the indoor images, and 36 subimages for the outdoor images. Each subimage's size is 20 by 20 pixels. The standard deviations of those subimages serve as inputs to each Bernoulli unit, i.e., each Bernoulli unit has a total of 48 inputs (and therefore, 48 weights) for the indoor image, and has a total of 36 inputs (36 weights) for the outdoor image. To learn the four selected *Phoenix* segmentation parameters, we need 20 Bernoulli units. So there is a total of 960 weights for indoor images, and 720 weights for outdoor images.

For the team of 20 Bernoulli units, the parameters α, γ, and δ are determined empirically, and they are kept constant for all images. In our experiments, $\alpha = 0.02$, $\gamma = 0.9$, and $\delta = 0.01$, *EB1* = 0.5, *EB2* = 0.4, *MaxGlobal, MaxLocal*, and *MaxSeg* are all set to 20. The threshold for matching confidence *Switch* = 0.6, and *Accept* = 0.8. Threshold used for extracting edges using Sobel operator is set at 200.

4.3.1 Results on Indoor and Outdoor Images

Figure 4.8 and 4.9 show the experimental results on the set of 12 indoor color images and the set of 12 outdoor color images. For each indoor image, the globally segmented

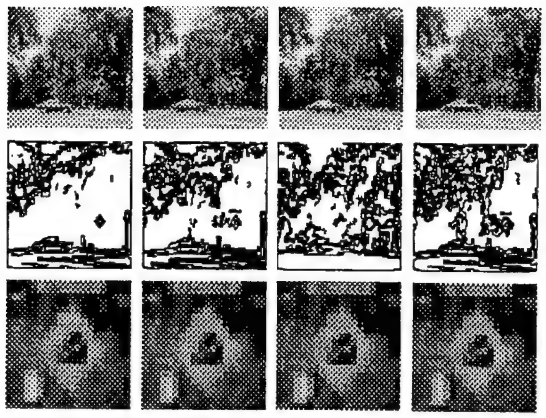


Figure 4.9: Row 1: input images; row 2, 3: corresponding segmented image and recognized object. For each input image, global segmentation evaluation, local segmentation evaluation for the selected object, and matching confidence are (0.59, 0.51, 0.82); (0.79, 0.57, 0.85); (0.85, 0.76, 0.88); (0.82, 0.53, 0.92). The learned *Phoenix* segmentation parameters *Hsmooth, Maxmin, Splitmin*, and *Height* after local learning process are (11 367 43 26); (11 259 23 46); (11 259 29 56); (9 276 31 46).

image using the set of learned parameters and the extracted object which has been finally recognized, are presented. For each set of images, the 12 images are taken sequentially. Except for the first image, the learning process for each image starts from the global segmentation parameters learned from all the previous images. For the first input image, the learning system is initialized using the unbiased RL algorithm. Usually, it takes less than 45 iterations to find a set of segmentation algorithm parameters which produces high edge-border coincidence. Figure 4.8 and 4.9 also show the global edge-border coincidence, local edge-border coincidence, model matching confidence, and the four learned segmentation parameters after local learning process for each input image.

Figure 4.10 shows the CPU time for the 12 indoor images and 12 outdoor images for five different runs, and the number of loops for each input image, which is the sum of all the loops involved in the global learning and local learning processes. These two curves show the learning capability of the system, i.e., the system uses less and less CPU time with experience to find a set of segmentation parameters and correctly recognizes the object. The number of learning loops decreases with the accumulation of experience.

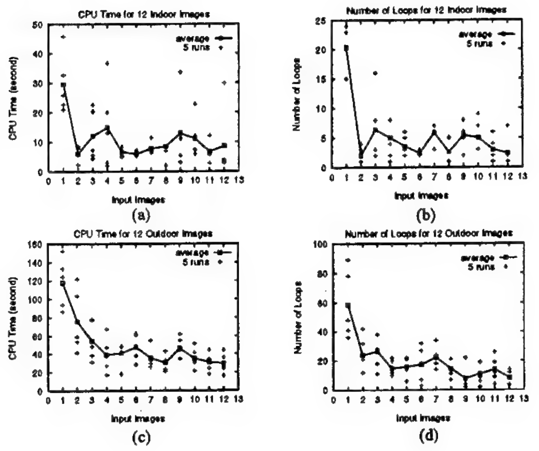


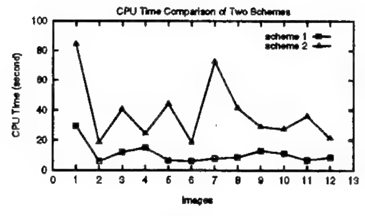
Figure 4.10: (a) CPU time for 5 different runs on 12 indoor images and the average; (b) Number of loops for 5 different runs on 12 indoor images and the average; (c) CPU time for 5 different runs on 12 outdoor images; (d) Number of loops for 5 different runs on 12 outdoor images.

4.3.2 Comparison of the Two Approaches

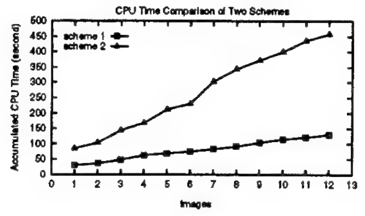
In this section we compare the performance of our system as shown in Figure 4.1 with the approach discussed in the paper by Peng and Bhanu [65]. We show the effect of incorporating segmentation evaluation using the edge-border coincidence into the learning system and the impact of global and local segmentations on model matching.

The key differences between the two methods are the introduction of the local segmentation process, the biasing of RL algorithm, and the use of edge-border coincidence as an evaluation of the segmentation performance during earlier stages of learning in order to reduce the computational expense stemming from model matching. The segmentation process alternates between the whole image and its subcomponents. The local segmentation is highly desirable when there are multiple targets or a single target at multiple locations with different local characteristics. It can dramatically improve the recognition performance. The biasing of RL algorithm reduces computational time as illustrated in Figure 4.6.

In the paper by Peng and Bhanu [65], the matching confidence is the only feedback that drives learning. Although it is undoubtedly the most reliable measure, it is relatively expensive to compute. Here the edge-border coincidence provides us with a cheap way to find a good point from which to begin the more expensive search for high matching confidence values. Figure 4.11 shows the comparison results of the two schemes: our scheme (scheme 1) and Peng and Bhanu's scheme (scheme 2). Although good initial estimates may



(a)



(b)

Figure 4.11: Comparison of two approaches: scheme 1–approach presented in this paper, scheme 2–Peng and Bhanu's approach (a) Comparison of the average CPU time of 5 different runs on 12 indoor images; (b) Comparison of the accumulated average CPU time of 5 different runs on 12 indoor images.

not always result in faster discovery of high matching confidence values, the edge-border coincidence seems to work well in practice for all the problems we have experimented.

4.4 Conclusions and Future Work

We have presented a proof-of-the-principle of a general approach for adaptive image segmentation and object recognition. The approach combines a domain independent simple measure for segmentation evaluation (edge-border coincidence) and domain dependent model matching confidence in a reinforcement learning framework in a systematic manner to accomplish robust image segmentation and object recognition simultaneously. Experimental results demonstrate that the approach is suitable for continuously adapting to normal changes encountered in real-world applications.

For adapting to the wide varity of images encountered in real-world applications, we can develop an autonomous gain control system which will allow the matching between different classes of images taken under significantly different weather conditions (sun, cloud, snow, rain) and adapt the parameters within each class of images. We use image context to divide the input images into several classes based on image properties and external conditions, such as time of the day, lighting condition, etc. [12]. When an image is presented, we use an image property measurement module and the available external information to find the stored information for this category of images, and start learning process from that set of parameters. This will overcome the problem of adapting to large variations between consecutive images.

The real significance of using a learning network to select segmentation parameters to optimize model matching performance is that interconnections within the network can enforce coordination of the choices made by the output units in order to concentrate the search in suspected high-payoff regions of the parameter space. A network that can coordinate the choices made by the output units should be able to generate certain combinations of bits with greater probability than if their individual components were selected independently. If the network operates in this way it should expect to find high matching confidence values much more quickly than without coordination. We plan to explore these issues in the future.

Chapter 5

Genetic Algorithm for Adaptive Image Segmentation

Image segmentation is an extremely important and difficult low-level task. The difficulty arises when the segmentation performance needs to be adapted to the changes in image quality which is affected by variations in environmental conditions, imaging devices, time of day, etc. In this Chapter, we describe an adaptive image segmentation system that incorporates a feedback loop consisting of a machine learning subsystem, an image segmentation algorithm, and an evaluation component which determines segmentation quality. The machine learning component is based on genetic adaptation and uses separately a pure genetic algorithm and a combination of genetic algorithm and hill climbing. We present experimental results which demonstrate learning and scalability of the technique with the number of parameters to adapt the segmentation performance in outdoor color imagery.

5.1 Introduction

Image segmentation is an old and difficult problem. It refers to the grouping of parts of an image that have "similar" image characteristics. All subsequent interpretation tasks including object detection, feature extraction, object recognition, and classification rely heavily on the quality of the segmentation process. The difficulty arises when the segmentation performance needs to be adapted to the changes in image quality. Image quality is affected by variations in environmental conditions, imaging devices, time of day, etc. Despite the large number of segmentation techniques presently available [26, 39], no general methods have been found that perform adequately across a diverse set of imagery, i.e., no segmentation algorithm can automatically generate an "ideal" segmentation result in one pass (or in an open loop manner) over a range of scenarios encountered in practical applications. Any technique, no matter how "sophisticated" it may be, will eventually yield poor performance if it cannot adapt to the variations in real-world scenes. The following are the key characteristics of the image segmentation problem:

- When presented with a new image, selecting the appropriate set of algorithm parameters is the key to effectively segmenting the image. Most segmentation techniques contain numerous **control parameters** which must be adjusted to obtain optimal performance, i.e., they are to be *learned.* The size of the parameter search space in these approaches can be prohibitively large, unless it is traversed in a highly efficient manner.
- The parameters within most segmentation algorithms typically interact in a complex, non-linear fashion, which makes it difficult or impossible to model the parameters' behavior in an algorithmic or rule-based fashion.
- The variations between images cause changes in the segmentation results, the objective function that represents segmentation quality varies from image to image. The search technique used to optimize the objective function must be able to adapt to these variations.
- The definition of the objective function itself can be a subject of debate because there are no universally accepted measures of image segmentation quality.

Hence, a need exists to apply an adaptive technique that can efficiently search the complex space of plausible parameter combinations and locate the values which yield optimal results. The approach should not be dependent on the particular application domain nor should it have to rely on detailed knowledge pertinent to the selected segmentation algorithm. Genetic algorithms (GA), which are designed to efficiently locate an approximate global maximum in a search space, have the attributes described above and show great promise in solving the parameter selection problem encountered in the image segmentation task.

The next section of this Chapter argues about the genetic algorithms as the appropriate optimization technique for the segmentation problem. Section 3 describes the adaptive image segmentation algorithm. Section 4 presents the experimental results on a sequence of outdoor images. Section 5 presents the adaptive segmentation results when we scale the number of parameters in a scheme that uses genetic algorithms and hill climbing. Finally, Section 6 provides the conclusions of this Chapter.

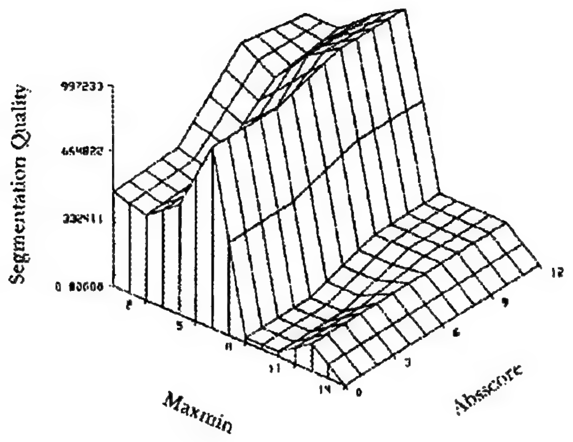


Figure 5.1: Segmentation quality surface.

5.2 Image Segmentation as an Optimization Problem

Fig. 1 provides an example of an objective function that is typical for the image segmentation process. The figure depicts an application in which only two segmentation parameters (*maxmin* and *absscore*) are being varied, and the corresponding segmentation quality obtained for any pair of algorithm parameters. Because the algorithm parameters interact in complex ways, the objective function is multimodal and presents problems for many commonly used optimization techniques. Further, since the surface is derived from an analysis of real-world imagery, it may be discontinuous, may contain significant amounts of noise, and cannot be described in closed form. The derivation of this surface will be described in Section 3, where we discuss the segmentation evaluation process.

The conclusion drawn from an analysis of many segmentation quality surfaces that we have examined is that we must utilize a highly effective search strategy which can withstand the breadth of performance requirements necessary for the image segmentation task.

Various commonly used search techniques for functional optimization exist. These include (a) exhaustive techniques (random walk, depth first, breadth first, enumerative), (b) calculus-based techniques (gradient methods, solving systems of equations), (c) partial knowledge techniques (hill climbing, beam search, best first, branch and bound, dynamic programming, A*), and (d) knowledge-based techniques (production rule systems, heuristic methods). The limitations of these methods are given in [12, 42, 95]. There are other search techniques such as genetic algorithms, simulated annealing and hybrid or integrated methods [12]. To address the characteristic of image segmentation problem as discussed earlier, we have selected genetic algorithms and hybrid methods for adaptive image segmentation.

5.2.1 Genetic Algorithms

Genetic algorithms were pioneered at the University of Michigan by John Holland and his associates [21, 31, 41]. The term genetic algorithm is derived from the fact that its operations are loosely based on the mechanics of genetic adaptation in biological systems. Genetic algorithms can be briefly characterized by three main concepts: a Darwinian notion of fitness or strength which determines an individual's likelihood of affecting future generations through reproduction; a reproduction operation which produces new individuals by combining selected members of the existing population; and genetic operators which create new offspring based on the structure of their parents.

A genetic algorithm maintains a constant-sized population of candidate solutions, known as individuals. The initial seed population from which the genetic process begins can be chosen randomly or on the basis of heuristics, if available for a given application. At each iteration, known as a generation, each individual is evaluated and recombined with others on the basis of its overall quality or fitness. The expected number of times an individual is selected for recombination is proportional to its fitness relative to the rest of the population. Intuitively, the high strength individuals selected for reproduction can be viewed as providers of "building blocks" from which new, higher strength offspring can be constructed. New individuals are created using two main genetic recombination operators known as crossover and mutation. Crossover operates by selecting a random location in the genetic string of the parents (crossover point) and concatenating the initial segment of one parent with the final segment of the second parent to create a new child. A second child is simultaneously generated using the remaining segments of the two parents. The string segments provided by each parent are the building blocks of the genetic algorithm. Mutation provides for occasional disturbances in the crossover operation by inverting one or more genetic elements during reproduction. This operation insures diversity in the genetic strings over long periods of time and prevents stagnation in the convergence of the optimization technique.

The individuals in the population are typically represented using a binary notation to promote efficiency and application independence of the genetic operations. Holland [41] provides evidence that a binary coding of the genetic information may be the optimal representation. Other characteristics of the genetic operators remain implementation dependent, such as whether both of the new structures obtained from crossover are retained, whether the parents themselves survive, and which other knowledge structures are replaced if the population size is to remain constant. In addition, issues such as the size of the population, crossover rate, mutation rate, generation gap, and selection strategy have been shown to affect the efficiency with which a genetic algorithm operates [34].

The inherent power of a genetic algorithm lies in its ability to exploit, in a highly efficient manner, information about a large number of individuals. By allocating more reproductive occurrences to above average individuals, the overall net affect is an upward shift in the population's average fitness. Since the overall average moves upward over time, the genetic algorithm is a "global force" which shifts attention to productive regions (groups of highly fit individuals) in the search space. However, since the population is distributed throughout the search space, genetic algorithms effectively minimize the problem of converging to local maxima.

To date, genetic algorithms have been applied to a wide diversity of problems. They have been used in combinatorial optimization [35], gas pipeline operations [30, 32] and machine learning [42]. With regards to computer vision applications, Mandava et. al [51] have used genetic algorithms for image registration, Gillies [29], and Roth and Levine [77] for feature extraction, and Ravichandran [71] for object recognition.

5.3 Genetic Learning for Adaptive Image Segmentation

Genetic algorithms can be used in several different ways to provide an adaptive behavior within a computer vision system [12]. The simplest approach is to allow the genetic system to modify a set of control parameters that affect the output of an existing computer vision program. By monitoring the quality of the resulting program output, the genetic system can dynamically change the parameters to achieve the best performance. In this paper, we have adopted this strategy for adaptive image segmentation.

The block diagram of our approach is shown in Fig. 2. After acquiring an input image, the system analyzes the image characteristics and passes this information, in conjunction with the observed external variables, to the genetic learning component. Using this data, the genetic learning system selects an appropriate parameter combination, which is passed to the image segmentation process. After the image has been segmented, the results are evaluated. If the quality of segmentation ("fitness") is acceptable, an update to long-term population is made. If the quality is unacceptable, the process of new parameter selection, segmentation and evaluation continues until a segmentation result of acceptable quality is produced, or the termination criteria are satisfied.

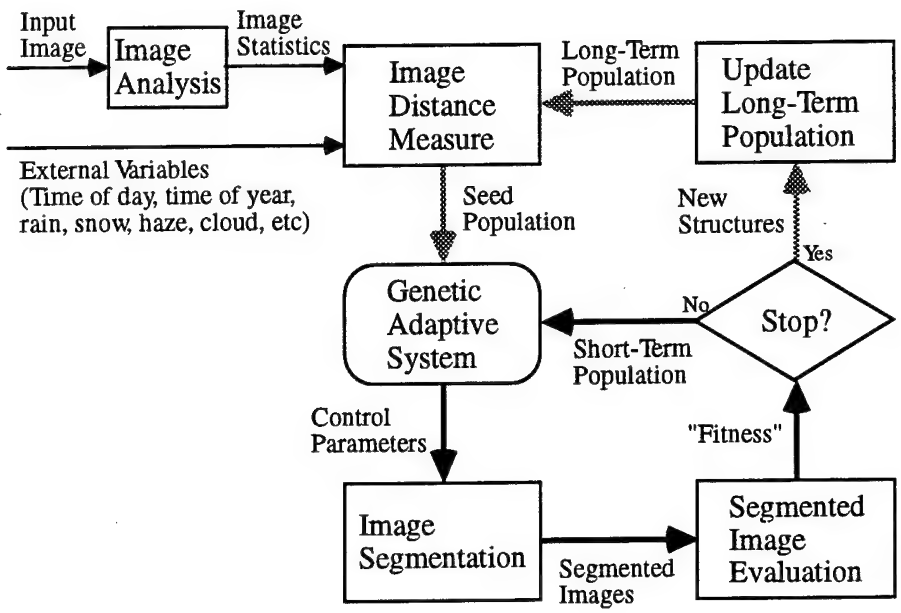


Figure 5.2: Adaptive image segmentation system.

5.3.1 Image Characteristics

A set of characteristics of the image is obtained by computing specific properties of the image itself as well as by observing the environmental conditions in which the image was acquired. Each type of information encapsulates knowledge that can be used to determine a set of appropriate starting points for the parameter adaptation process. For the experiments described here, we compute twelve first order properties for each color component (red, green, and blue) of the image. These features include mean, variance, skewness, kurtosis, energy, entropy, x intensity centroid, y intensity centroid, maximum peak height, maximum peak location, interval set score, and interval set size [46, 84]. The last two features measure histogram properties used directly by the *PHOENIX* segmentation algorithm used in this research and provide useful image similarity information. Since we use a gray scale image to compute edge information and object contrast during the evaluation process, we also compute the twelve features for the Y (luminance component) image as well. Combining the image characteristic data from these four components yields a list of 48 elements. In addition, we utilize two external variables, time of day and weather conditions to characterize

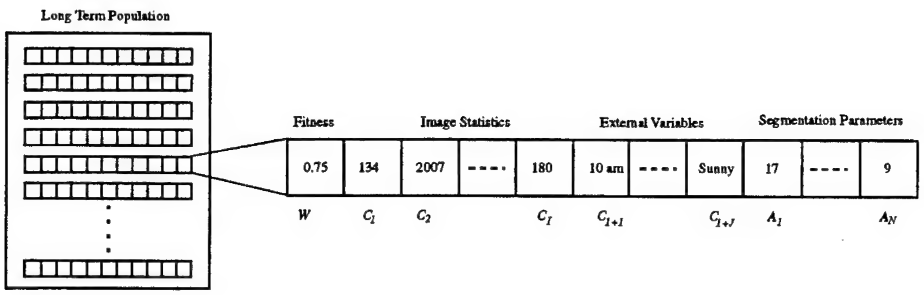


Figure 5.3: Representation of a knowledge structure used by the genetic learning system. The image characteristics (image statistics and external variables), segmentation parameters, and the image quality or fitness of the parameter set are stored in each structure.

each image. The external variables are represented symbolically in the list structure (e.g., time = 9am, 10am, etc. and weather conditions = sunny, cloudy, hazy, etc). The distances between these values are computed symbolically when measuring image similarity. The two external variables are added to the list to create an image characteristic list of 50 elements. The representation of an individual knowledge structure of the genetic population is shown in Fig. 3, where I is the number of image statistics, J is the number of external variables and N is the number of segmentation parameters.

5.3.2 Genetic Learning System

Once the image statistics and external variables have been obtained, the genetic learning component uses this information to select an initial set of segmentation algorithm parameters. A knowledge-based system is used to represent the image characteristics and the associated segmentation parameters. The image statistics and external variables shown in Fig. 3 form the condition portion of the knowledge structure, C_1 through C_{I+J}, while the segmentation parameters indicate the actions, A_1 through A_N, of the knowledge structure. The fitness, W, which ranges in value from 0.0 to 1.0, measures the quality of the segmentation parameter set. Note that only the fitness value and the action portion of the knowledge structure are subject to genetic adaptation; the conditions remain fixed for the life of the knowledge structure.

When a new image is provided to the genetic learning system, the process begins by comparing the image characteristics of the new image (Fig. 2) with the knowledge structures in the long-term population (also called global population, Fig. 3). The long-term population represents the accumulated knowledge of the adaptive system obtained through previous segmentation experience. The algorithm computes a ranked list of individuals in the population that have characteristics similar to the new image. Ranking is based on the normalized Euclidean distance between the image characteristic values as well as the fitness of the knowledge structure. The normalized distance between images A and B is computed using

$$dist_{AB} = \sum_{i=1}^{I+J} W_i \left| \frac{C_{iA} - C_{iMIN}}{C_{iMAX} - C_{iMIN}} - \frac{C_{iB} - C_{iMIN}}{C_{iMAX} - C_{iMIN}} \right| ,$$

where C_{iMIN} is the minimum value of the ith numeric or symbolic feature in the global population, C_{iMAX} is the maximum value of the ith feature in the global population, and W_i is the weight attached to the ith feature. For the results presented in this paper, the ranges are normalized and the W_i values have been set to 1 so that each feature contributes equally to the distance calculation.

When the distance between an image and several members of the global population are the same (e.g., if a previous image contributed multiple individuals to the global population), fitness values are used to select the best individuals from the population. Temporary copies of the highest ranked individuals are used to create the initial or seed population for the new image.

Once the initial or seed population is available, the genetic adaptation cycle begins. (The seed population is the same as the initial population, when the genetic algorithm begins its search operation.) The segmentation parameter set in each member of the seed population is used to process the image. The quality of the segmented results for each parameter set is then evaluated. If the maximum segmentation quality for the current population is above a predefined threshold of acceptance or other stopping criteria are satisfied, the cycle terminates and the high quality members of the current image population are used to update the global population. Less fit members of the global population are discarded in favor of higher strength individuals obtained from processing the current image. In this manner, the system is able to extend the knowledge of the adaptive segmentation system by incorporating new experience into the knowledge database.

Alternatively, if after segmenting and evaluating the performance of the current or local (also called short-term) population, the system has not achieved acceptable segmentation quality and any other termination criteria are not satisfied, the genetic recombination operators are applied to the members of the current population. The crossover and mutation operators are applied to the high strength individuals in the population, creating a new set of offspring which will theoretically yield better performance [12, 41]. The new population is supplied back to the image segmentation process, where the cycle begins again. Each pass through the loop (segmentation-evaluation-recombination) is known as a generation.

The cycle shown continues until the maximum fitness achieved at the end of a generation exceeds some threshold or other termination criteria are satisfied. The global population is updated and the system is then ready to process a new image.

5.3.3 Segmentation Algorithm

Since we are working with color imagery in our experiments, we have selected the *PHOENIX* segmentation algorithm developed at Carnegie-Mellon University and SRI International [46, 59, 84]. The *PHOENIX* algorithm is a recursive region splitting technique. An input image typically has red, green, and blue image planes, although monochrome images, texture planes, and other pixel-oriented data may also be used. Each of the data planes is called a feature or feature plane. The algorithm recursively splits nonuniform regions in the image into smaller subregions on the basis of a peak/valley analysis of the histograms of the red, green, and blue image components simultaneously. Segmentation begins with the entire image, considered to be a single region, based on histogram and spatial analyses. If the initial segmentation fails, the program terminates; otherwise, the program fetches each of the new regions in turn and attempts to segment them. This process terminates when the recursive segmentation reaches a predefined depth, or when all the regions have been segmented as finely as various user-specified parameters permit.

PHOENIX contains seventeen different control parameters [46], fourteen of which are used to control the thresholds and termination conditions of the algorithm. There are about 10^{40} conceivable parameter combinations using these fourteen values. For the outdoor image sequence that we have used, these parameters can be divided into three groups according to their effect on segmentation results.

Group I: Essential *PHOENIX* Parameters.

Parameter (default)	*Description*	*Range*
Hsmooth (9)	The width of the averaging window used to smooth each feature histogram.	$1 - 100$
Maxmin (160)	The minimum acceptable ratio of apex height to higher shoulder.	$100 - 10^4$

Group II: Important *PHOENIX* Parameters.

Parameter (default)	*Description*	*Range*
Absscore (70)	The lowest interval set score that will be passed to the threshold phase.	0 – 1000
Splitmin (4)	Direct manipulation of the segmentation queue, for which fetched regions are to be segmented further.	1 – 200
Noise (10)	The size of the largest area that is to be considered noise.	$0 - 10^4$
Height (20)	The minimum acceptable apex height as a percentage of the second highest apex.	0 – 100

Group III: Less important *PHOENIX* parameters

The rest of the parameters have relatively much less influence on the segmentation result.

To minimize the problem complexity, four parameters have been selected for GA to search for the combination that gives best segmentation result using *PHOENIX*. Thirty two values are sampled for each of these four parameters. This results in a search space whose size is about one million. The parameters are shown in Table 1, together with the formula by which they are sampled, and the associated test range for each. In Section 4, we will present results using the first two parameters (*hsmooth* and *maxmin*). In Section 5, we show scaling results when we adapt all the four parameters.

5.3.4 Segmentation Evaluation

After the image segmentation process has been completed by the *PHOENIX* algorithm, we must measure the overall quality of the segmented image. There are a large number of segmentation quality measures [7] that have been developed in the past, although none has achieved widespread acceptance as a universal measure of segmentation quality. In order to overcome the drawbacks of using only a single quality measure, we have incorporated an evaluation technique that uses five different quality measures to determine the overall fitness for a particular parameter set. In the following, boundary pixels refer to the pixels along the borders of the segmented regions, while the edges obtained after applying an edge operator are called edge pixels. The five segmentation quality measures that we have selected are,

1. *Edge-Border Coincidence*: Measures the overlap of the region borders in the image acquired from the segmentation algorithm relative to the edges found using an edge operator. In this quality measure, we use the Sobel operator to compute the necessary edge information. The original, unthinned Sobel edge image is used to maximize overlap between the segmented image and the edge image. Edge-border coincidence is defined as follows (refer to Fig. 4(a)).

   Let E be the set of pixels extracted by the edge operator after thresholding and S be the set of pixels found on the region boundaries obtained from the segmentation algorithm:

   $$E = \{p_1, p_2, \cdots, p_E\} = \{(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), \cdots, (x_{pE}, y_{pE})\} \quad and$$
   $$S = \{q_1, q_2, \cdots, q_S\} = \{(x_{q1}, y_{q1}), (x_{q2}, y_{q2}), \cdots, (x_{qS}, y_{qS}\}, \quad then$$

   $$\text{Edge-border Coincidence} = \frac{n(E \cap S)}{n(E)}$$
   $$E \cap S = \{(x_k, y_k), k = 1, \cdots, m, where (x_y, y_k) \in E \text{ and } S\}, \quad and$$
   $$n(A) = \text{the number of elements in set A.}$$

2. *Boundary Consistency*: Similar to edge-border coincidence, except that region borders which do not exactly overlap edges can be matched with each other. In addition, region borders which do not match with any edges are used to penalize the segmentation quality. The Roberts edge operator is used to obtain the required edge information. As with the edge-border coincidence measure, the Roberts edge image is not thinned to maximize the overlap between images. Boundary consistency is computed in the following manner (see Fig. 4(b)).

   The first step is to find neighboring pixel pairs in the region boundary and edge results. For each pixel in the segmented image region boundary results, S, a neighboring pixel in the edge image, E, that is within a distance of d_{max} is sought. A reward for locating a neighbor of the ith boundary pixel is computed using

   $$R_i = \frac{d_{max} - d_i}{d_{max}},$$

   where $d_{max} = 10$, and d_i = the distance to the nearest edge pixel.

   Thus, if the pixels had overlapped, $R_i = (10 - 0)/10 = 1$. Pixels that do not directly overlap contribute a reward value that is inversely related to their distance from each other. As matching pairs of pixels are identified, they are removed from the region boundary and edge images (S and E). The total reward for all matching pixel pairs is obtained using

$$R_{TOTAL} = \sum_i R_i$$

Once all neighboring pixel pairs have been removed from E and S, the remaining (i.e., non-overlapping and non-neighboring) pixels correspond to the difference between the two images. The average number of these pixels is used to compute a penalty

$$P = \frac{n(\text{all remaining pixels in } E \text{ and } S)}{2}.$$

Finally, since the value of boundary discrepancy must be positive, we define an intermediate value, M, as $M = (R_{TOTAL} - P)/n(E)$, then

$$\text{Boundary Consistency} = M, \quad \text{if } M \geq 0, \quad \text{and zero otherwise.}$$

3. *Pixel Classification*: This measure is based on the number of object pixels classified as background pixels and the number of background pixels classified as object pixels. Let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image (see Fig. 4(c)). Formally, we have

$$G = \{p_1, p_2, \cdots, p_A\} = \{(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), \cdots, (x_{pA}, y_{pA})\} \quad and$$
$$R = \{q_1, q_2, \cdots, q_B\} = \{(x_{q1}, y_{q1}), (x_{q2}, y_{q2}), \cdots, (x_{qB}, y_{qB}\}.$$

Since pixel classification must be positive, we define the intermediate value N as follows

$$N = 1 - \left[\frac{(n(G) - n(G \cap R)) + (n(R) - n(G \cap R))}{n(G)}\right]$$
$$\text{where } G \cap R = \{(x_k, y_k), k = 1, \cdots, m, \quad \text{where } (x_k, y_k) \in G \text{ and } R\}$$

Using the value of N, pixel classification can then be computed as

$$\text{Pixel Classification} = N, \quad \text{if } N \geq 0, \quad \text{and zero otherwise.}$$

4. *Object Overlap*: Measures the area of intersection between the object region in the groundtruth image and the segmented image, divided by the object region. As defined in the pixel classification quality measure, let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image (Fig. 4(d)). Object overlap can be computed as

$$\text{Object Overlap} = \frac{n(G \cap R)}{n(G)}$$
$$\text{where } G \cap R = \{(x_k, y_k), k = 1, \cdots, m, \quad \text{where } (x_k, y_k) \in G \text{ and } R\}$$

5. *Object Contrast*: Measures the contrast between the object and the background in the segmented image, relative to the object contrast in the ground-truth image. Let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image, as shown in Fig. 4(a). In addition, we define a bounding box (X and Y) for each object region in these images. These boxes are obtained by enlarging the size of the minimum bounding rectangle for each object (G and R) by 5 pixels on each side. The pixels in regions X and Y include all pixels inside these enlarged boxes with the exception of the pixels inside the G and R object regions. We compute the average intensity for each of the four regions (G, R, X, and Y) using the equation $I_L = \sum_{j=1}^{L_{max}} I(j)/L_{max}$, where $I(j)$ is the intensity of the jth pixel in some region L and L_{max} is the total number of pixels in region L. The contrast of the object in the groundtruth image, C_{GT}, and the contrast of the object in the segmented image, C_{SI}, can be computed using

$$C_{GT} = \left|\frac{I_G - I_X}{I_G}\right|, \quad C_{SI} = \left|\frac{I_R - I_Y}{I_R}\right|.$$

The object contrast quality measure is then computed as

$$\begin{aligned} \text{Object Contrast} &= \frac{C_{SI}}{C_{GT}}, \quad \text{if } C_{GT} \geq C_{SI} \\ &= \frac{C_{GT}}{C_{SI}}, \quad \text{if } C_{GT} < C_{SI}. \end{aligned}$$

The maximum and minimum values for each of the five segmentation quality measures are 1.0 and 0.0, respectively. The first two quality measures are *global* measures since they evaluate the segmentation quality of the whole image with respect to edge information. Conversely, the last three quality measures are *local* measures since they only evaluate the segmentation quality for the object regions of interest in the image. When an object is broken up into smaller parts during the segmentation process, only the largest region which overlaps the actual object in the image is used in computing the local quality measures. In the experiments described in this chapter, we combine the five quality measures into a single, scalar measure of segmentation quality using a weighted sum approach. Each of the five measures is given equal weighting in the weighted sum. Elsewhere we have investigated a more complex vector evaluation approach that provides multidimensional feedback on segmentation quality [12, 13].

5.4 Segmentation Results

5.4.1 Segmentation Using Genetic Algorithm

The adaptive image segmentation consists of the following steps:

```
1.      Compute the image statistics.
2.      Generate an initial population.
3.      Segment the image using initial parameters.
4.      Compute the segmentation quality measures.
5.      WHILE not <stopping conditions> DO
5a.         select individuals using the reproduction operator
5b.         generate new population using the crossover
            and mutation operators
5c.         segment the image using new parameters
5d.         compute the segmentation quality measures
        END
6.      Update the knowledge base using the new knowledge structures.
```

We have tested the performance of the adaptive image segmentation system on a time sequence of outdoor images. The outdoor image database consisted of twenty frames captured using a JVC GXF700U color video camera. The images were collected approximately every 15 minutes over a 4 hour period. A representative subset of these images is shown in Fig. 5. The original images were digitized to be 480×480 pixels in size but were subsequently subsampled (average of 4×4 pixel neighborhood) to produce 120×120 pixel images for the segmentation experiments. Weather conditions in our image database varied from bright sun to overcast skies. The changing environmental conditions caused by movement of the sun also created varying object highlights, moving shadows, and many subtle contrast changes between the objects in the image. Also, the colors of most objects in the image are subdued. The auto-iris mechanism in the camera was functioning, which causes a similar appearance of the background foliage throughout the image sequence. Even with the auto-iris capability built into the camera, there was still a wide variation in image characteristics across the image sequence. This variation required the use of an adaptive segmentation approach to compensate for these changes.

The car in the image is the object of interest for the pixel classification, object overlap, and object contrast segmentation quality measures. The groundtruth image for the car was obtained by manual segmentation of Frame 1 only for the image sequence. The Sobel and Roberts edge operator results, which are used in the computation of the edge-border coincidence and boundary consistency measures respectively, are obtained from the gray scale image (Y component of the YIQ image set) for each frame [14]. For the results presented in this section, the *maxmin* and *hsmooth* parameters of the *PHOENIX* algorithm were used to control the segmentation quality and the segmentation quality surfaces were defined for preselected ranges of these two parameters as shown in Table 1. All the parameters that were not optimized were set at the default *PHOENIX* parameter values. These parameters remain fixed throughout all the experiments. By selecting 32 discrete values (5 bits of resolution) for each of these parameter ranges, the search space contained 1024 different parameter combinations. Fig. 6 presents the five individual segmentation quality surfaces and the combined surface for Frame 1 of the database. Notice that the surfaces are complex and hence, would pose significant problems to traditional optimization techniques.

The genetic component used a local or seed population size of 10, long-term population size of 100, a crossover rate of 0.8, and mutation rate of 0.01. A crossover rate of 0.8 indicates that, on average, 8 out of 10 members of the population will be selected for recombination during each generation. The mutation rate of 0.01 implies that on average, 1 out of 100 bits is mutated during the crossover operation to insure diversity in the local population. The stopping criterion for the genetic process contains three tests. First, since the global maximum for each segmentation quality surface was known *a priori* (the entire surface was precomputed to evaluate results), the first test is the location of a parameter combination that produces quality of 95% or higher. In experiments where the entire surface is not precomputed, this test would be discarded. Second, the process terminates if three consecutive generations produce a decrease in the average population fitness for the local population. Third, if five consecutive generations fail to produce a new maximum value for the average population fitness, the genetic process terminates. If any one of these three conditions is met, the processing of the current image is stopped and the maximum segmentation quality currently in the local population is reported.

Numerous experiments [12, 14] were performed for training and testing to measure the optimization capabilities of the genetic algorithm and to evaluate the reduction in effort achieved by utilizing previous segmentation experience. In the following we present some of these results.

5.4.2 Performance Comparison with Other Techniques

Since there are no other known adaptive segmentation techniques with a learning capability in both the computer vision and neural networks fields to compare our system with, we measured the performance of the adaptive image segmentation system relative to the set of default *PHOENIX* segmentation parameters [46, 84] and a traditional optimization approach. The default parameters have been suggested after extensive amounts of testing by researchers who developed the *PHOENIX* algorithm [46]. The parameters for the traditional approach are obtained by manually optimizing the segmentation algorithm on the first image in the database and then utilizing that parameter set for the remainder of the experiments. This approach to segmentation quality optimization is currently a standard practice in state-of-the-art computer vision systems. Fig. 7 illustrates the quality of the segmentation results for Frames 1 and 11 using the default parameters and the traditional approach and contrasts this performance with our adaptive segmentation technique. By comparing the extracted car region in each of these images, as well as the overall segmentation of the entire image, it is clear that the adaptive segmentation results are superior to the other methods. For the 20 frames the average segmentation quality for the adaptive segmentation technique is 95.8%. In contrast, the performance of the default parameters is only 55.6% while the traditional approach has a 63.2% accuracy. The size of the search space in these experiments is 1024, since each of the two *PHOENIX* parameters are represented using 5 bits. The price paid for achieving consistent higher quality of segmentation is the average number of times (2.5) one has to go through the genetic loop. Thus, only 2.4% of the search space is explored to achieve the global maximum. Many additional tests, including the comparison with random walk approach, are performed to demonstrate the effectiveness of the reproduction and crossover operators [12].

5.4.3 Demonstration of Learning Behavior

The above experiments were conducted in a parallel fashion, i.e., all training and all testing was performed without the aid of previous segmentation experience. Although the testing experiments used the knowledge acquired during training, the tests were still performed in parallel. None of the segmentation experience obtained during testing was applied to subsequent testing images. The following multiple day experiment shows that experience can be used to improve the segmentation quality over time. The test simulates a four day scenario where the frequency of image acquisition decreases to approximately one hour. The order of the images in this test is 1, 5, 9, 12, 16, 20, 3, 7, 11, 14, 18, 2, 6, 10, 13, 17, 4, 8, 15, 19. Each group of images in the sequence of Frames (1, 5, 9, 12, 16, 20), (3, 7, 11, 14, 18), (2, 6, 10, 13, 17), or (4, 8, 15, 19) was designed to represent a collection of images acquired on a different day.

The genetic population of the first frame in the image sequence was randomly selected. Once the segmentation performance for that frame was optimized by the genetic algorithm, the final population from that image was used to create the initial global population. This global population was then used to select the seed population for subsequent frames in the image sequence. The global population size was set to 100 for these experiments to insure a diversity of segmentation experience in the population. While the size of the global population remained below 100 members (prior to processing 10 frames), the final populations for each image were merely added to the current global population. After the size of the global population reached 100 individuals, the final populations from each successive image had to compete with the current members of the global population. This competition was based on the fitness of the individuals; highly fit members of a new local population replaced less fit members of the global population, thus keeping the size of the global population constant. Fig. 8 presents the performance results achieved by the adaptive image segmentation system during each of the three sequential tests. The images in the first "day" (frames 1, 5, 9, 12, 16, 20) show a continually decreasing level of computational effort. When the second sequence (frames 3, 7, 11, 14, 18) is encountered, the effort increases temporarily as the adaptive process fills in the knowledge gaps present as a result of the differences between the images in each sequence. The image sequence for the third "day" (frames 2, 6, 10, 13, 17) was handled with almost no effort by the genetic learning. Finally, the fourth image sequence (frames 4, 8, 15, 19) requires no effort by the genetic learning at all; each image is optimized by the information stored in the global population. Twelve of the twenty frames in this test were optimized using the global population.

5.5 Scaling the Number of Parameters

For the results presented in Section 4, we selected only two (*hsmooth* and *maxmin*) parameters of the *PHOENIX* algorithm. In this section, we present details when we select four parameters (*hsmooth*, *maxmin*, *splitmin* and *height*) for adaptive image segmentation. In this case the size of the search space is about 1 million. Table 1 shows the parameter values. As the number of segmentation parameters for adaptation increases, the number of points to be visited on the surface will also increase. However, genetic algorithms offer a number of advantages over other search techniques. These include parallel search from a set of points with the *expectation* of achieving the global maximum. Unlike the Hough transform, which is essentially an exhaustive search technique commonly used in Computer Vision, it is expected that the genetic algorithm will visit only a small percentage of the search space to find an adequate solution that is sufficiently close to the global maximum.

5.5.1 Search Space and GA Control Mechanism

Visualization of the Search Space: Visualization of the search space allows one to understand its complexity—the number and distribution of local peaks and the location of global maximum. But this 5-dimensional space (four parameters plus the fitness or quality of image segmentation) is difficult to be visualized with traditional methods. So we project this 5-dimensional data into a 4-dimensional space by slicing it into 32 pieces along the

Height axis. Fig. 9 shows the 3-D volume representation of this 4-dimensional data using the brick and slice visualization technique, where the x, y, z axes are *maxmin*, *hsmooth*, and *splitmin* respectively, and the color associated with each point represents the combined segmentation quality for a given parameter set. Blue color represents segmentation quality of 0, while the red color represents 100% quality.

GA Control Mechanism: GA learning requires 3 operations: selection, crossover, and mutation. In our approach, a chromosome is formed by combining the 4 segmentation parameters together. Using our method of crossover point selection, the ordering of these parameters within the chromosome does not affect the search process. Tests are carried out to select the best control parameters for GA, which include the number of crossover points, crossover rate, method of selection, population size, and quality threshold. The results given below are averaged over 1000 independent tests.

Crossover Rate: Table 2 shows the number of segmentations that are needed for frame 1 for different crossover rates. The threshold for minimum acceptable segmentation quality is 95%, population size varies from 50 to 200. We can see that a lower crossover rate leads to smaller number of total segmentations.

Population Size and Number of Crossover Points: Table 3 shows the number of segmentations required for different population sizes and crossover points. The threshold for acceptance of segmentation quality is 95% and the crossover rate is set at 80%. From the results we can see that using more crossover points and larger population size, the total number of required segmentations can be reduced. This experiment also showed that the total number of segmentations will not reduce further when population size is greater than 500. A complete scenario for crossover operation using four points is shown in Fig. 10.

Segmentation Quality Threshold: Table 4 shows how different thresholds for minimum acceptable segmentation quality affect the total number of required segmentations. The difference is not significant between 90% and 95% because these segmentation qualities are quite close.

The results presented for Frame 1 in Tables 2-4 show that the number of points that are visited on the surface varies from 0.9% to 0.3% for 95% quality of segmentation. In the best case only 0.28% of the search space is visited to achieve 99.89% (threshold is 95%) quality of segmentation.

5.5.2 Genetic Algorithms and Hill Climbing

Integrated search techniques have the potential for improved performance over single optimization techniques since these can exploit the strengths of the individual approaches in a cooperative manner [1, 13]. One such scheme which we describe in this section combines a global search technique (genetic algorithms) with a specialized local search technique (hill climbing). Hill climbing methods are not suitable for optimization of multimodal objective functions, such as the segmentation quality surfaces, since they only lead to local extrema. The integrated scheme provides performance improvements over the genetic algorithm alone by taking advantage of both the genetic algorithm's global search ability and the hill climbing's local convergence ability. In a sense, the genetic algorithm first finds the hills and the hill climber climbs them.

The search through a space of parameter values using hill climbing consists of the following steps: (1) Select a starting point; (2) Take a step in each of the fixed set of directions; (3) Move to the best alternative found; and (4) Repeat until a point is reached that is higher than all of its adjacent points. An algorithmic description of the hill climbing process is given below.

Table 5.1: *PHOENIX* parameters used for adaptive image segmentation.

Parameter	*Sampling Formula*	*Test Range*
Hsmooth: $hsindex \in [0:31]$	$hsmooth = 1 + 2 \cdot hsindex$	1 — 63
Maxmin: $mmindex \in [0:31]$	$ep = \log(100) + 0.05 \cdot mmindex$ $maxmin = \exp(ep) + 0.5$	100 — 471
Splitmin: $smindex \in [0:31]$	$splitmin = 9 + 2 \cdot smindex$	9 — 71
Height: $htindex \in [0:31]$	$height = 4 + 2 \cdot htindex$	4 — 66

Table 5.2: Number of segmentations under varying population size and crossover rate. The threshold for minimum acceptable segmentation quality was set at 95%.

Population	Crossover Rate	2-Point Crossover
50	80%	9439
	50%	6077
100	80%	5805
	50%	4675
200	80%	7548
	50%	5068

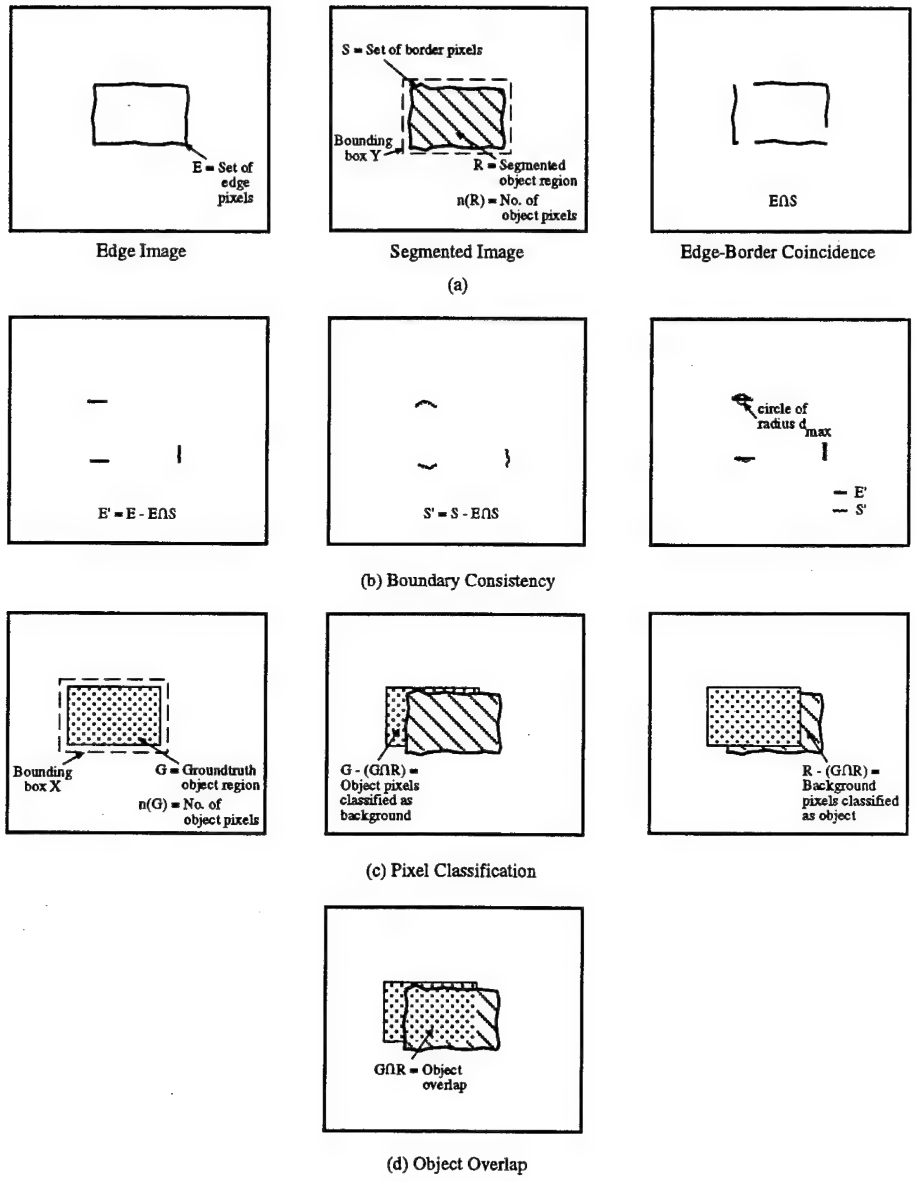
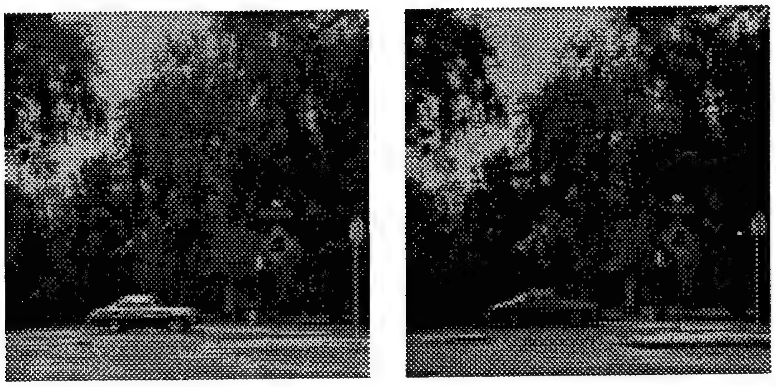


Figure 5.4: Illustration for the quality measures used in the adaptive image segmentation system. (a) Edge-border coincidence, (b) Boundary consistency, (c) Pixel classification, (d) Object overlap. Object contrast is defined by using the symbols shown in the center figure in (a) and the left most figure in (c).



(a) Frame 1 (b) Frame 11

Figure 5.5: Sample outdoor images used for adaptive segmentation experiments.

Table 5.3: Number of segmentations under varying population size and crossover points (Segmentation quality threshold = 95% , Crossover rate = 80%).

Population	1-Point Crossover	2-Point Crossover	4-Point Crossover
10	7102	6553	5941
100	4960	5805	5528
200	4131	3939	3900
500	3575	3332	2878

Table 5.4: Number of segmentations under varying threshold (Population = 500, Crossover rate = 80%).

Threshold	1-Point Crossover	2-Point Crossover	4-Point Crossover
95%	3575	3332	2878
90%	2943	2788	2325

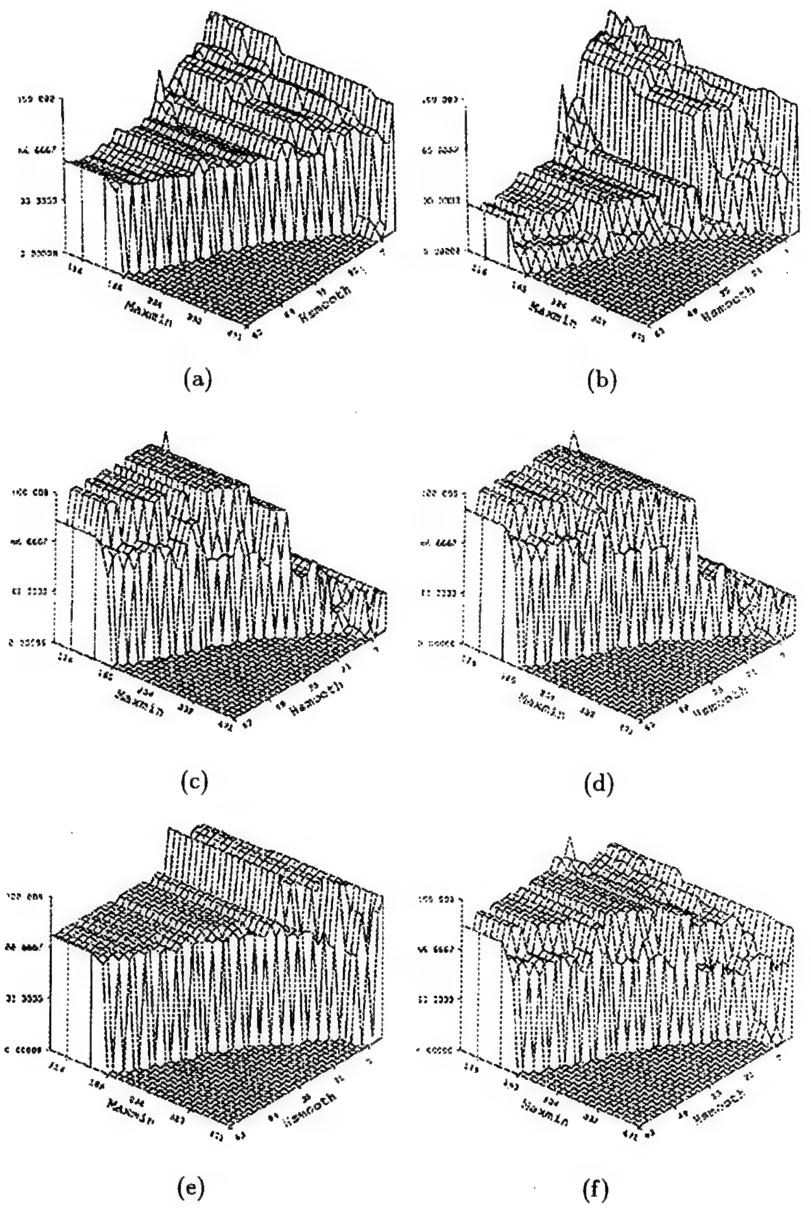


Figure 5.6: Segmentation quality surfaces for Frame 1. (a) Edge-border Coincidence, (b) Boundary Consistency, (c) Pixel Classification, (d) Object Overlap, (e) Object Contrast, (f) Combined Segmentation Quality.

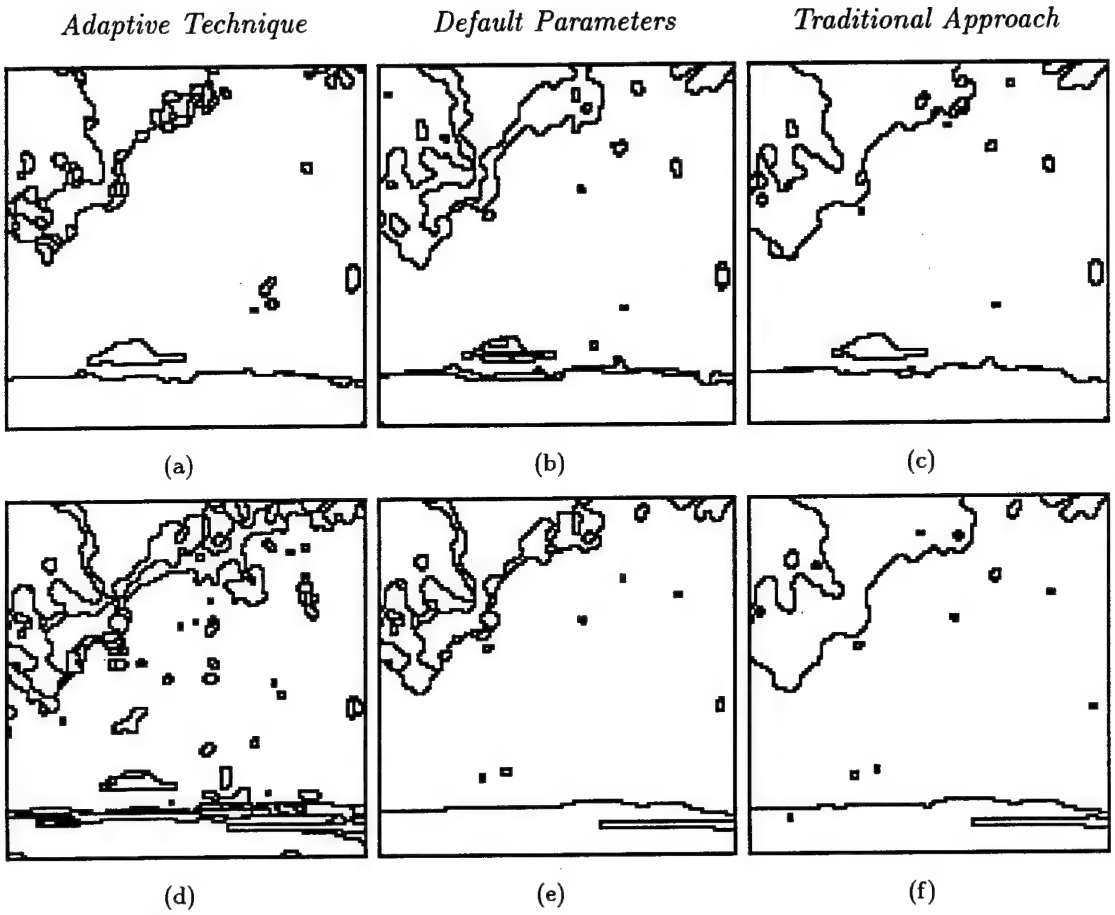


Figure 5.7: Segmentation of Frame 1 (a–c) and Frame 11 (d–f) for the adaptive technique, default parameters, and the traditional approach.

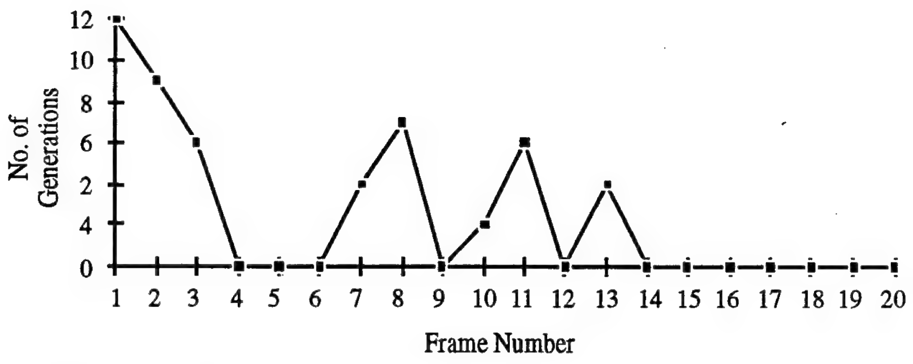
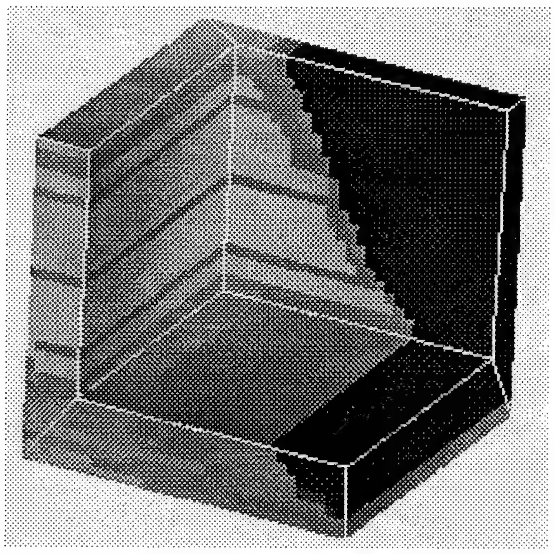
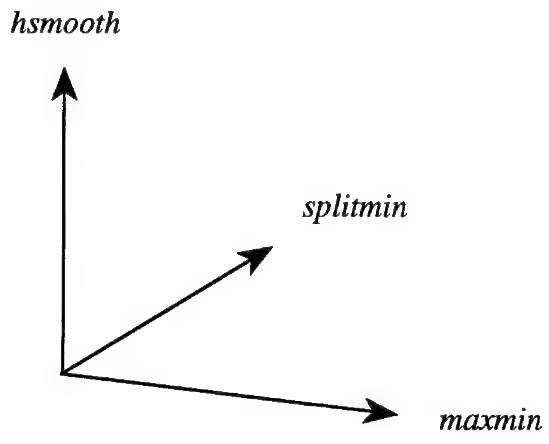


Figure 5.8: Performance of the adaptive image segmentation system for a multiple day sequential test.

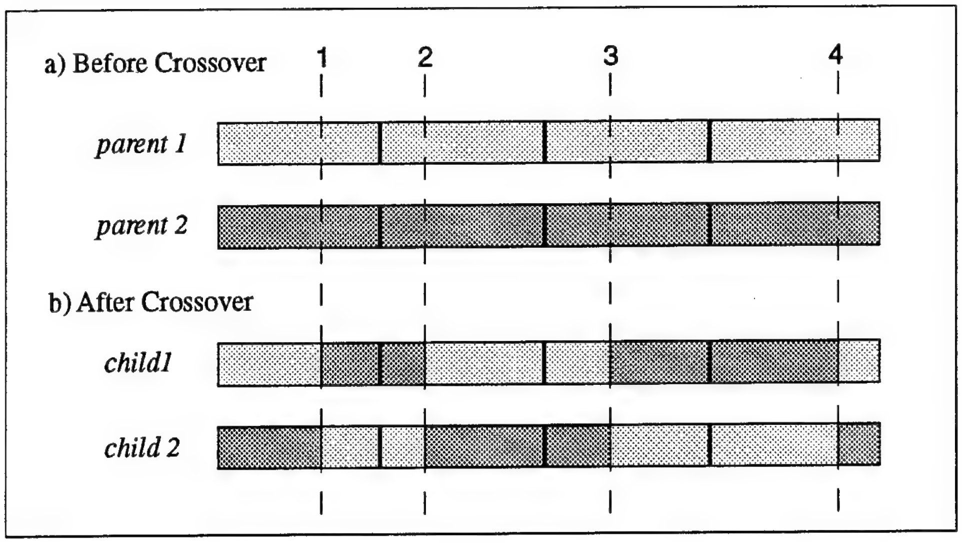


(a) Projection with *height* = 10

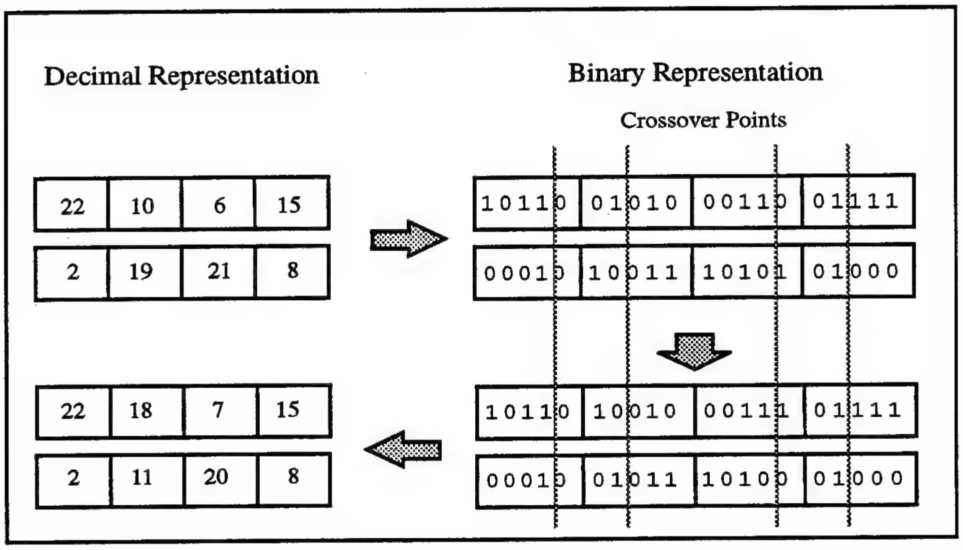


(b) Coordinate axes

Figure 5.9: Volume representation of segmentation parameter search space. (a) The original 5-dimensional data (hsmooth, splitmin, maxmin, height, segmentation quality) is projected along height axis, where the color represents the fitness or segmentation quality value. (b) The coordinate system.



(a)



(b)

Figure 5.10: Genetic algorithm crossover operation. (a) Scheme for doing 4-point crossover with each chromosome containing four parameters. (b) A complete scenario for one crossover operation.

```
1a.    Select a point $x_c$ at random.
1b.    Evaluate the criterion function, i.e., obtain $V(x_c)$.
2a.    Identify points $x_1, \cdots, x_n$ adjacent to $x_c$.
2b.    Evaluate the criterion function, i.e., obtain $V(x_1), \cdots, V(x_n)$.
3.     Let $V(x_m)$ be the maximum of $V(x_i)$ for $i = 1, \cdots, n$.
3a.    If $V(x_m) > V(x_c)$ then
           set $x_c = x_m, V(x_c) = V(x_m)$.
           goto Step 2.
3b.    Otherwise, stop.
```

In this algorithm, a set of points that are "adjacent" to a certain point can be defined in two ways. First, it can denote the set of points that are a Euclidean distance apart from the given point. Thus, the adjacent points are located in the neighborhood of the given point. Second, "adjacent" points can denote the set of points that are unit Hamming distance apart from the given point pair. Each point in this set differs by only one bit value from the given point in binary representation of points. It defines the set of points with varying step size from the given point. The set of Hamming adjacent points was used in this research. Hamming adjacent points have an advantage over Euclidean adjacent points in our implementation because all the segmentation parameter values are represented as binary strings when using the GA. The set of Hamming adjacent points also represents the set of points which can be generated by a genetic mutation operator from the given point.

A conventional hill climbing approach, as described above, finds the largest $V(x_m)$ from $V(x_i), i = 1, \cdots, n$, and the search moves to its corresponding point, x_m. For a space of n adjacent points, it requires n function evaluations to make each move. To reduce the cost of evaluating all the adjacent points before making each move, the approach is designed to try alternatives only until an uphill move is found. The first uphill move is undertaken without checking whether there are other (higher) possible moves. After the hill climbing process has examined all the adjacent points by flipping each bit in the binary representation of the current point, in turn, without finding an uphill move, the current point is taken as a local maximum. The algorithmic description of the hill climbing process used in the search scheme is as follows:

```
1.     Select a starting point $x_c$ with fitness value $V(x_c)$ from the
       genetic population.
2.     Set $i = 0$.
3.     Set $j = i$.
4a.    Generate an adjacent point $x_a$ by flipping the $i$th bit in $x_c$.
4b.    Obtain $V(x_a)$. Set $i = (i + 1) \bmod n$.
5.     If $V(x_a) > V(x_c)$ then
```

```
            set $x_c = x_a$.
            goto Step 3.
Else if $i < j$ then
            goto Step 4.
Otherwise, pass the control to the GA.
```

5.5.3 Experimental Results

There are several possibilities in which genetic algorithms and hill climbing can be used. In one case the control moves back and forth between GA and hill climbing [12, 13]. In this approach when GA finds a new maximum, hill climbing is used to keep climbing until local maximum or termination condition is satisfied. If a local maximum is found then GA is again used to find a new maximum. For the experiments presented in this Chapter this approach is used for the first frame only. Specifically, the integrated technique used is given below:

1. Perform GA and hill climbing search for frame 1 using a population size of 10 (chosen from available hardware consideration) and 4 point crossover operation with a crossover rate of 0.8 (same as in Section 4). The goal here is to use small population size to achieve the desired segmentation quality with a minimum number of segmentations.

2. For frame 2 to frame 20 perform hill climbing with accumulated knowledge structures. The parameter set generated from previous frames is used to hill climb. The best result obtained for the current frame is kept as a new knowledge structure and added to the parameter set for hill climbing for the next frame.

After we are done with frame 20, a total of 29 knowledge structures are accumulated, with 19 of them generated by hill climbing.

The experimental results for frame 1 are shown in Table 5. The results show that for 95% threshold for image segmentation quality, the technique helps to reduce the required number of segmentations by almost half. For low segmentation quality threshold (90%), this effect is not dramatic.

Fig. 11 summarizes the performance of the technique for frames 1 to 20, and compares it with the performance of the default parameter set of the *PHOENIX* algorithm [46]. The performance corresponds to the parameter set in the population that has the highest fitness. The average performance improvement for the technique over the default parameter set is about 50%, performance improvement over the technique that uses the parameter set

Table 5.5: Performance comparison between pure GA and GA with hill climbing (crossover points = 4, crossover rate = 80%, mutation rate ≈ 1%).

Population = 10	Genetic w/o hill climbing	Genetic with hill climbing
Threshold = 95%	5941	3340
Threshold = 90%	1720	1631

generated by GA plus hill climbing learning for frame 1 only (no subsequent hill climbing) is also significant. This shows that learning from frame 1 does provide a good starting point for hill climbing for subsequent frames. The maximum improvement over the default parameter set shown in Fig. 11 is 107.8%.

Fig. 12 shows the sample segmentation results obtained by using the default parameter set and the parameter set generated by the technique. Using the default parameter set, it is seen that the car does not show up at all in the segmentation result of frame 16, but the corresponding result using GA and hill climbing is quite good. The overall results show that by combining genetic search and hill climbing techniques the performance improvement is significant when the search space is large.

5.6 Conclusions

The goal of this research was to perform adaptive image segmentation and evaluate the convergence properties of the closed-loop system using outdoor data. In this Chapter we have provided sample results. Using the outdoor data we have shown in [12, 13, 14] that the performance improvement provided by the adaptive system was consistently greater than 30% over the traditional approach or the default segmentation parameters [46, 84].

The adaptive image segmentation system can make use of any segmentation technique that can be controlled through parameter changes. The adaptive segmentation system is only as robust as the segmentation algorithm that is employed. It may be possible to keep multiple segmentation algorithms available and let the genetic process itself dynamically select the appropriate algorithm based on image characteristics. Further, it is possible to define various evaluation criteria which can be automatically selected and optimized in a complete vision system. In a complete computer vision system, the segmentation evaluation component can be replaced by the object recognition component(for example, see [64]). In

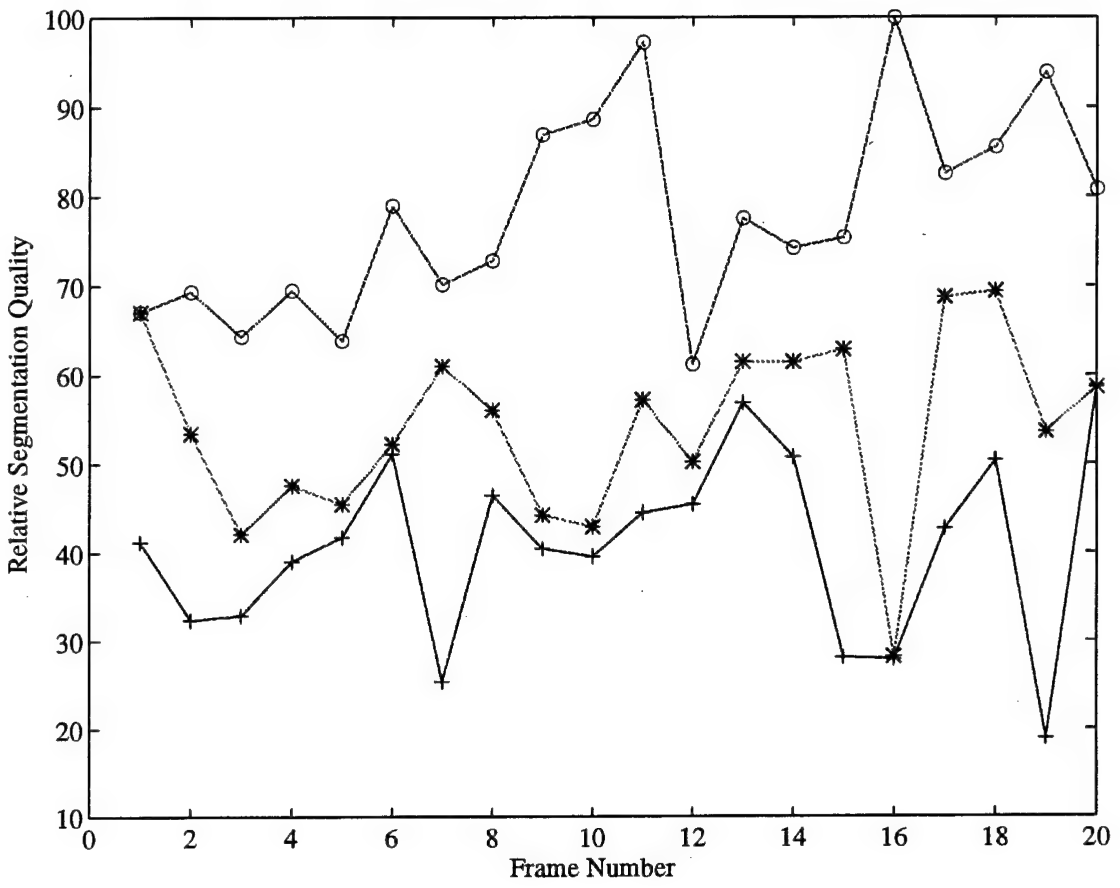


Figure 5.11: Performance comparison for techniques based on (a) default parameters (+), (b) GA plus hill climbing to generate the best parameter set for frame 1 only (*), and (c) integrated technique, (parameter set generated for frame 1 in the same manner as in (b) and hill climbing for all subsequent frames (o)).

our adaptive image segmentation system, the focus is the image segmentation component. Therefore, we supplied the manually generated groundtruth image to the segmentation evaluation component and used local and global measures. Elsewhere, we have optimized both global and local measures in a multi-objective optimization framework [13]. In the future we plan to use a data set with dramatic environmental variations and we will utilize several segmentation algorithms. Ultimately, we will incorporate the adaptive segmentation component into a learning integrated object recognition system.

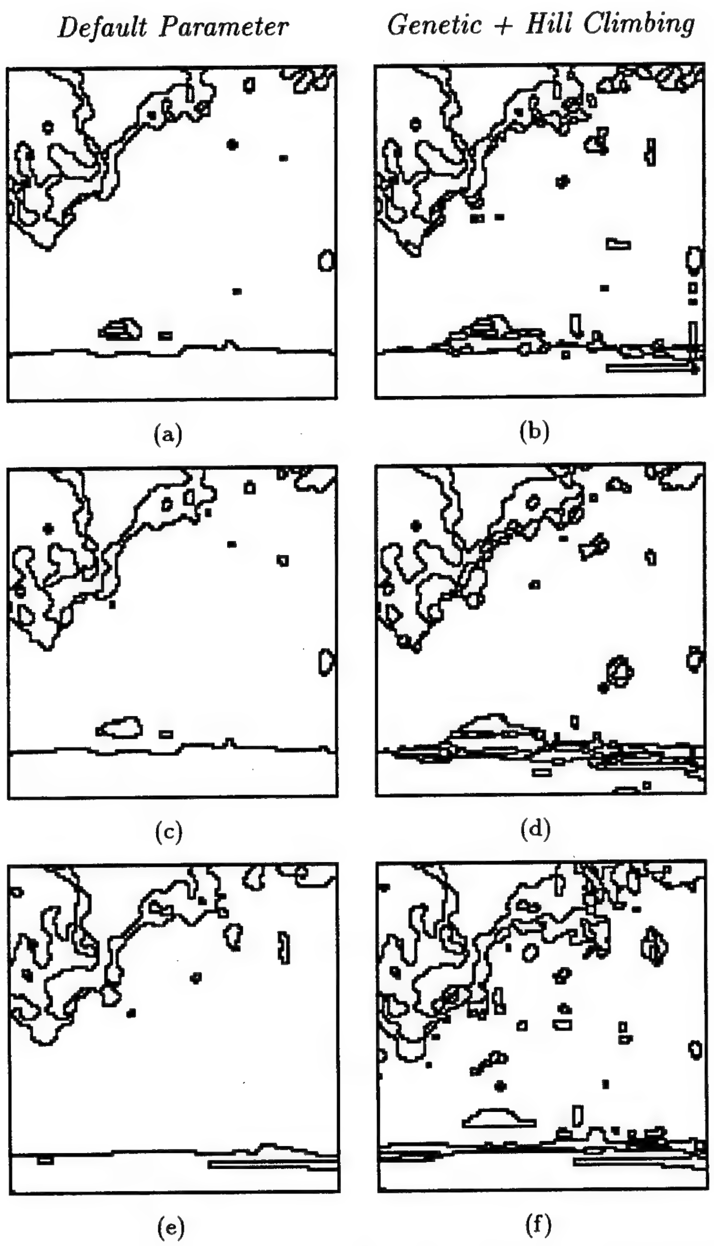


Figure 5.12: Segmentation performance comparison using default and learned parameters. (a) and (b) Frame2, (c) and (d) Frame 3, (e) and (f) Frame 16.

Chapter 6

Integrating Context with Clutter Models for Target Detection

For an automatic target detection and recognition (ATD/R) system operating in a cluttered environment, it is important to develop models not only for man-made targets but also for various background clutters. Because of the high complexity of natural backgrounds, our approach to build the clutter models is based on learning from real examples. The contextual parameters, which describe the environmental conditions for each training example, are used in a reinforcement learning paradigm to improve the learned clutter models and enhance target detection performance under multi-scenario situations. We present experimental results using second generation infrared imagery.

6.1 Introduction

Automatic Target Detection/Recognition (ATD/R) is a challenging application for the general techniques developed by image understanding communities [11]. There are several reasons that contribute to this challenge: (a) a target may appear in many different backgrounds and it tends to be mixed up with its surroundings, (b) signatures of a target strongly depends upon the background surrounding the target and environmental conditions, and (c) signatures of a target are generally not repeatable. As a result, early stage image segmentation for extracting the target from the background is generally unreliable. Since the ATD/R algorithms are commonly used in a sequential manner, any target we fail to detect during the detection stage will be lost forever. In the detection stage, it is desired to single out every suspicious target area (region-of-interest) in the image, even at the cost that it may include some false target areas. Then it is the responsibility of the following recognition stage to verify the identity of each real target and to filter out the false targets. An ideal ATD/R system is the one that (1) does not miss any potential target area in the detection stage, and (2) does not verify any non-target area as target in the recognition stage.

To achieve the goal of high detection probability and simultaneous low false-alarm rate, we have developed an ATD/R strategy called Background Model Aided Target Detection and Recognition (BMATDR). The main idea of BMATDR is to use explicit background clutter models, as well as target models, throughout the ATD/R process. The clutter models are represented by a bank of statistical models that are constructed using various multiresolution and other image feature groups through a self-organizing learning process [74, 75]. Although these statistical models characterize the background clutter using a variety of information present in a training image, they do not utilize any non-imagery contextual information associated with each training image. Because we want to successfully detect targets under multi-scenario situations, and the image metrics commonly used to characterize images do not correlate well with performance of an ATD/R system, it is essential to integrate the contextual information with clutter models for target detection. In this paper, we present a reinforcement learning based approach that improves the target detection performance of background clutter models by using the non-imagery information. The non-imagery contextual information accompanying a training image is represented by a set of *context parameters*, and a certain setting of these contextual parameters is referred to as a *contextual condition.* We present results using 40 second generation infrared images.

6.2 Learning Background Models via Self-Organizing Maps

In recent years, two streams of approaches have been developed by ATR researchers to characterize the natural background in infrared images. The first one is using heat transfer equations to model the thermal behaviors of different materials. The second stream of approaches focuses on the image features rather than the thermal-physical meaning behind these images. In our previous work, we followed the second approach, and developed a image feature-based background clutter modeling system [74]. Several image feature groups were developed based on multiresolution analysis and local geometric analysis [9]. All the image features developed are computed from rectangular regions in an image, which we refer to as *feature cells.* Based on each feature group, which consists of not more than three feature values, a statistical model was learned from training images and represented by a $2D$ self-organizing map [75].

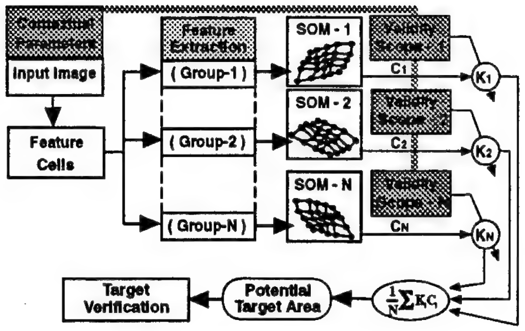


Figure 6.1: Learning background clutter models for target detection.

Since natural backgrounds can occur in a wide variety, background characterization must rely on multiple features. To efficiently use the available features, we need a proper representation to hold all the information together. One way to attack such a problem is to organize all the features into a high dimensional feature vector (i.e. long feature vector) and classify the backgrounds based on the position of the vector in some high dimensional feature space. The other way is based on short feature vectors. The key ideas behind this later scheme are: we need to understand the physical meaning of each feature and put each feature in a group of features that have closely related physical meanings. For a given background we will have a collection of simple (i.e. low dimensional) models. We refer to such a collection of models as a *Background Model Bank (BMB)*, and each model in this bank as a *BMB member* (SOM-1, $\cdots$, SOM-n in Figure 6.1). In the following discussions, background model and clutter model will be used interchangeably to refer to a BMB member.

6.3 The Classification Criterion

Since the target detection process has been formulated as a 2-class classification problem (natural background vs. man-made target) in our approach, a classification criterion is needed to label a testing *feature cell* according to the learned background model. The criterion used in our experiments is based on the computation of two confidence values: $Conf_B$, the confidence value for a testing feature cell to be background and $Conf_T$, the confidence value for it to be man-made target. These two confidence values are computed by comparing the *Four Neighbor Average Distance (FNAD)* of a testing feature cell to $\bar{R}_P$ which is the average FNAD of all the positive training examples, and $\bar{R}_N$ which is the average

$FNAD$ of all the negative training examples. To compute the $FNAD$ of a feature cell, the feature vector is first projected into the learned background model which is represented by a self-organizing map. The hitting neuron is found next, and the four neighbors of this hitting neuron are identified. The FNAD can then be calculated according to

$$FNAD = \frac{\sum_{i=1}^{4} r_i}{4} \tag{6.1}$$

where r_i is the Euclidean distance between the testing feature vector and one of the 4-Neighbor neurons. Figure 6.2 shows the projection of the testing feature vector into the background model SOM. From Equation (6.1) we have

$$\bar{R}_P = \frac{\sum_{n=1}^{N_P} \sum_{i=1}^{4} r_i^n}{4N_P} \tag{6.2}$$

$$\bar{R}_N = \frac{\sum_{n=1}^{N_N} \sum_{i=1}^{4} r_i^n}{4N_N} \tag{6.3}$$

where N_P is the number of positive training examples used in constructing the background model, and N_N is the number of near-misses used. The two confidence values are

$$Conf_B = \max\left(1, \frac{4\bar{R}_P}{\sum_{i=1}^{4} r_i}\right) \tag{6.4}$$

$$Conf_T = \max\left(1, \frac{\sum_{i=1}^{4} r_i}{4\bar{R}_N}\right) \tag{6.5}$$

Given $Conf_B$ and $Conf_T$, the classification of the testing feature cell is obtained as,

$$l = \begin{cases} Background & \text{if } Conf_B > Conf_T \\ Target & \text{Otherwise} \end{cases} \tag{6.6}$$

The classification confidence C_C is assigned the value of $Conf_B$ or $Conf_T$ accordingly.

6.4 Reinforcement of Clutter Models Using Contextual Information

It can be imagined that different features may have different sensitivities to a certain contextual parameter, e.g. in FLIR images, the mean and standard deviation of image gray values are more sensitive to the air temperature than the Gabor transform amplitude features [74] which tend to find out periodic patterns within a local image region. To be practically

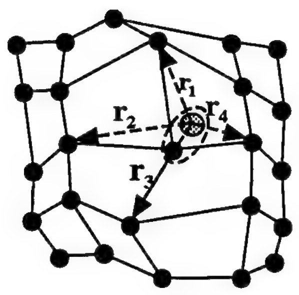


Figure 6.2: Projecting a testing feature vector into a clutter model.

applicable, an ATD/R system must be able to detect targets under different contextual conditions, i.e. under multi-scenario situations. One way to achieve this goal is to use a learning technique to associate contextual parameters with the performance of each feature group in the background model bank. The rationale behind this association is that if a feature group can effectively detect man-made objects under a given contextual condition, then it tends to be effective for images taken under similar contextual conditions. Since a human supervisor cannot provide any assistance to the ATD/R system in finding this association, except telling the system whether it is doing a "good job" with respect to a specific testing image, the most suitable learning technique for this task is the *reinforcement learning* scheme. Figure 6.3 shows how this reinforcement learning subsystem fits into the background modeling process.

If a feature group has a good performance under a certain contextual condition, its detection result deserves a larger weight in the background model bank under similar contextual conditions. In other words, the *context - performance* relationship can be replaced by a *context - weight* relationship, which is more compliant for being integrated into an automatic learning system. To facilitate the discussion, we first define the following terms, which will be used to formulate the following stochastic reinforcement learning algorithm. The superscript $i (i = 1, 2, \cdots, n_C)$ refers to the i-th contextual parameter available to our BMATDR system. The subscript $j (j = 1, 2, \cdots, n_F)$ refers to the j-th feature group in the background model bank.

- *Contextual Parameter* (c^i) is a scalar that quantifies a specific aspect of a contextual condition, it can be defined over continuous or discrete values.
- *Contextual vector* (**C**) is a vector with each element being c^i, a *contextual parameter.*
- Weight Vector (**W**) is a real value vector with each element being w_j, the weight of

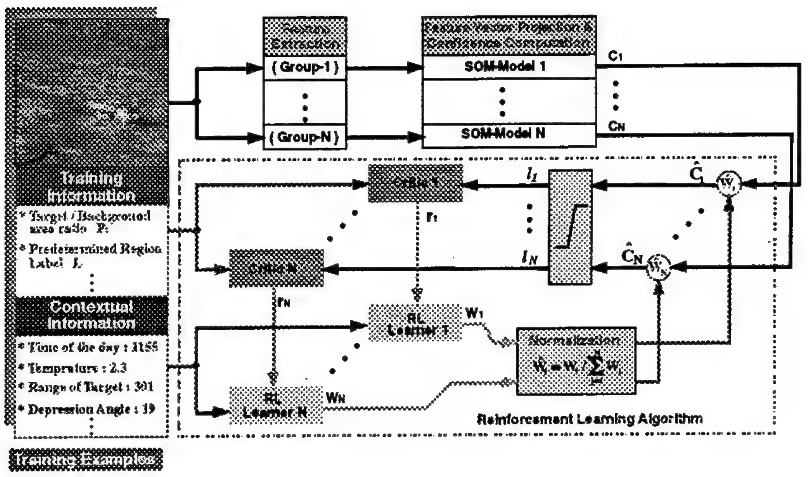


Figure 6.3: Context reinforced clutter modeling process.

the j-th feature group.

Our learning problem can then be defined as: Given a set of training images that cover the entire range of available contextual parameters, with the background model bank having been built as a collection of SOM's, associate with each BMB member SOM_j a stochastic transform function T_j, such that $w_j = T_j(\mathbf{C})$. T_j is stochastic because the Context($\mathbf{C}$) - Weight($\mathbf{W}$) relationship cannot be described by a deterministic function, and there are always exceptional cases due to the high complexity of the real world.

6.4.1 Stochastic Reinforcement Learning Algorithm

The reinforcement learning algorithm we selected for learning the *context - performance* relationship of each BMB member is shown in Figure 6.4. It is based on the *stochastic real valued reinforcement learning* algorithm, which is developed by Gullapalli for training a single *actor* using reinforcement as feedback [36, 37]. This algorithm allows the system to learn outputs that take on real values. Since the performance of a feature group is best described as a real number, normally between 0 and 1, with 1(0) representing the best (worst) performance, this algorithm meets our requirement very well. However, since we want to use this algorithm to cooperate the actions of n_F BMB members to achieve a better performance in target detection, we need to make changes to the algorithm, so that it can handle multiple *actors.*

In the stochastic reinforcement learning algorithm, T_j is implemented as a random number generator according to the normal distribution. The mean μ_j, and standard deviation σ_j are determined by two internal vectors, $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$ according to the following formula:

$$\mu_{j,n} = \mathbf{\Phi}^t_{j,n} \cdot \mathbf{C}_n \tag{6.7}$$

where n denotes the n-th iteration of the learning process.

$$\sigma_{j,n} = 1 - \hat{r}_{j,n} \tag{6.8}$$

where $\hat{r}_{j,n}$ is the estimated reinforcement feedback for the j-th feature group after n iterations of learning. It can be estimated using the following formula:

$$\hat{r}_{j,n} = 1 - f(\mathbf{\Psi}^t_j \cdot \mathbf{C}_n) \tag{6.9}$$

where function $f(\cdot)$ often takes the form of

$$f(a) = \frac{1}{1 + e^{-a}} \tag{6.10}$$

which maps the real line onto the interval $(0, 1)$. Once the two parameters (μ_j, σ_j) are available, the weight for the j-th feature group can be computed by passing μ_j and σ_j to a random number generator:

$$w_j \sim N(\mu_j, \sigma_j) \tag{6.11}$$

So, in the learning system, each context-to-weight transform function T_j is actually "remembered" as two *internal vectors*, $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$. Starting with randomly selected initial values, these two internal vectors learn to represent the Context($\mathbf{C}$) - Weight($\mathbf{W}$) relationship by updating themselves according to the following formula.

$$\mathbf{\Phi}_{j,n+1} = \mathbf{\Phi}_{j,n} + \sigma_{j,n}(r_{j,n} - \hat{r}_{j,n})(w_{j,n} - \mu_{j,n})\mathbf{C}_n \tag{6.12}$$

$$\mathbf{\Psi}_{j,n+1} = \mathbf{\Psi}_{j,n} + \rho_n(r_{j,n} - \hat{r}_{j,n})\mathbf{C}_n \tag{6.13}$$

where

$$r_{j,n} = g(\mathbf{P}_{j,n})$$

is the reinforcement provided by a critic function $g(\cdot)$ for the the j-th feature group in the n-th detection trial. Vector $\mathbf{P}$ is the vector for the detection result. It can be used by the *critic* to judge the performance of the system after the detection trial. Figure 6.4 shows the data flow path of the modified stochastic reinforcement learning algorithm. Gullapalli has provided a convergence proof of the single actor reinforcement learning algorithm [37]. The convergence of the multiple actor case still lacks a proof, it is one of our future research topic.

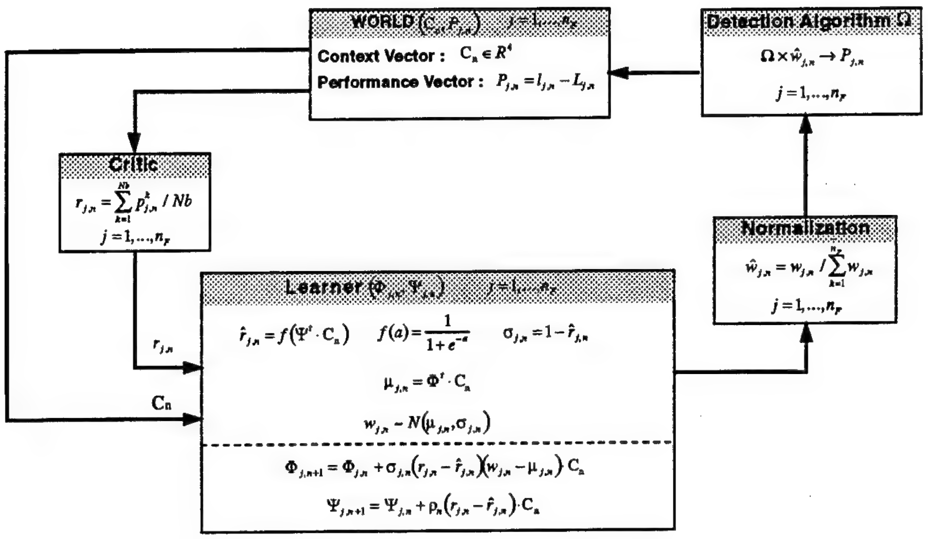

WORLD $(C_n, P_{j,n})$ $j=1,\dots,n_r$
Context Vector : $C_n \in R^4$
Performance Vector : $P_{j,n} = l_{j,n} - L_{j,n}$
Detection Algorithm Ω
$\Omega \times \hat{w}_{j,n} \to P_{j,n}$
$j=1,\dots,n_r$
Critic
$r_{j,n} = \sum_{k=1}^{Nb} p_{j,n}^k / Nb$
$j=1,\dots,n_r$
Normalization
$\hat{w}_{j,n} = w_{j,n} / \sum_{k=1}^{n_r} w_{j,n}$
$j=1,\dots,n_r$
Learner $(\Phi_{j,n}, \Psi_{j,n})$ $j=1,\dots,n_r$
$\hat{r}_{j,n} = f(\Psi^t \cdot C_n)$
$f(a) = \frac{1}{1+e^{-a}}$
$\sigma_{j,n} = 1 - \hat{r}_{j,n}$
$\mu_{j,n} = \Phi^t \cdot C_n$
$w_{j,n} \sim N(\mu_{j,n}, \sigma_{j,n})$
$\Phi_{j,n+1} = \Phi_{j,n} + \sigma_{j,n}(r_{j,n} - \hat{r}_{j,n})(w_{j,n} - \mu_{j,n}) \cdot C_n$
$\Psi_{j,n+1} = \Psi_{j,n} + \rho_n(r_{j,n} - \hat{r}_{j,n}) \cdot C_n$
$r_{j,n}$
C_n

Figure 6.4: The stochastic reinforcement learning algorithm.

6.4.2 Implementation Concerns

To utilize this complex learning scheme to solve the previously defined Context(**C**) - Weight(**W**) problem, we have to make several implementation decisions:

1. *Selection of contextual parameters*:
   For infrared images, There are many contextual parameters available. Sherman et al. categorized 41 such parameters into five classes — background parameters, target parameters, platform dynamics, atmospherics and sensor parameters [86]. Obviously, it is difficult to deal with 41 contextual parameters at the same time without organizing context in some hierarchical manner. One simple practical approach is to select a subset of important contextual parameters from the available context. In our implementation, we selected 4 parameters to form the contextual vector. These parameters are:

   - t : *Time of the day.*
   - d : *Depression angle.*
   - s : *Range to the target.*

- p : *Air temperature.*

In order to make the inner product in equations (6.7) and (6.8) meaningful, we used relative values of the contextual parameters in constructing the contextual vector. The relative value of d, for example, is $\frac{d}{d_{max}-d_{min}}$, where d_{max} and d_{min} are the maximum and minimum depression angle encountered in the training images.

2. *Performance vector* **P**:
Since all our features [74] are region based features, given a testing image, the image is first divided into feature cells based on the *range-to-target* information. The detection result is a label vector **l**, with each of its element being the label of a feature cell. The label of the top-left feature cell is the first element in **l**, and the label of rest feature cells are entered in a row-first manner. The easiest way to describe the performance of the detection is to compare **l** with the label vector **L** given by the learning supervisor. Thus, the performance vector **P** can simply be $\mathbf{P} = \mathbf{l} - \mathbf{L}$.

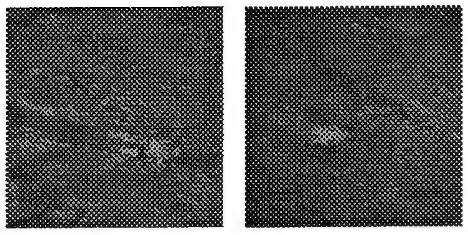
3. *Selection of the critic function*:
Since we are dealing with a two class classification problem, both **l** and **L** can be a bit vector. A simple metric for the detection performance is obtained by examining the number of bits being set to 1 in **P**. So, the reinforcement feedback can be computed as follows:

$$r_{j,n} = \sum_{k=1}^{Nb} p_{j,n}^{k}/Nb \tag{6.14}$$

where Nb is the total number of feature cells within the testing image. $p_{j,n}^{k}$ denotes the k-th element of vector $\mathbf{P}_{j,n}$ which, in turn, describes the performance of the j-th feature group in the n-th detection trial.

6.5 Experimental Results

In this experiment, we compare the detection performances of two background model banks learned by using our near-miss injection SOM algorithm [75] and 40 FLIR images (20 images for training and the other 20 for testing). Two sample images are shown in Figure 6.5. In the *first* background model bank, no contextual information is involved. Thus all the BMB members (each represented by a 5×5 self-organizing map) in the background model bank are treated as equally important. Since we have five BMB members in our background model bank, the weight of each BMB member is 0.2. The *second* background model bank contains the same BMB members that constitute the first background model bank. In addition, contextual information is used to reinforce this background model bank. By



(a) 09$p3sa6r_0h$ (b) 09$p3sa6r_2h$

Figure 6.5: Two examples of the 40 FLIR images used in our experiments.

applying the stochastic reinforcement learning algorithm given in Section 6.4.1 for 200 iterations, a relationship is set up between the importance of each BMB member and the underlying contextual condition. This relationship is represented by the two internal vectors Φ and Ψ associated with each BMB member.

After the construction of these two background model banks, another 20 second generation FLIR images are used as testing images. For each testing image, an accompanied image is built by removing rows and columns, equal to one-half the size of the selected feature cell, from all the four sides. By using the *first* background model bank and the classification criterion given in Section 6.3, out of the 217 feature cells in the 20 testing images, we achieved a 100% detection rate and a 12% false alarm. These 20 testing images and their accompanied images are then classified by using the *second* background model bank. The detection rate obtained is 100% and false alarm decreased by 2%. The detection confidence values of the correctly classified feature cells in both experiments are shown in Figure 6.6(a). Figure 6.6(b) shows the confidence values of the misclassified feature cells in both experiments. From these two figures we can see that by reinforcing the background model bank using contextual information, the confidence values of the correctly classified feature cells increase, and the confidence values of the misclassified feature cells decrease. The final effect is an improved detection performance. In Figure 6.6(c), the result is presented in a different way. The confidence values of the misclassified feature cells are represented by negative values, as before, the confidence values of the correctly classified feature cells are still represented as positive values. It can be seen that the context reinforced background model makes the curve of confidence values shifting upward, which produces a better detection result. We expect that further reduction in false alarms are possible by increasing the size of the training image set, which would expose the background model bank to more contextual conditions. In our experiments, because of the availability of experimental im-

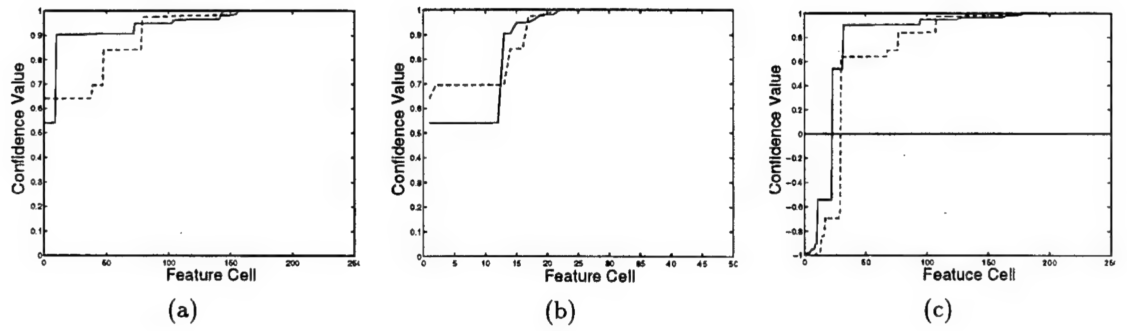


Figure 6.6: The improved detection performance after context reinforcement of the background model. Dashed lines show results of the background model bank not reinforced by the contextual information. Solid lines represent results of the context reinforced background model bank. (a) correctly classified feature cells (b) misclassified feature cells (c) all the feature cells in the testing images.

ages, the contextual conditions of some testing images are apparently different from those of the training images. This has limited the reduction in false alarms in our experiment.

6.6 Conclusions

By introducing learning capabilities into an ATD/R system, we can build a model for the complex natural background from real images and improve it as the system is trained with more examples. Contextual parameters which hold non-imagery information of the training examples are used to enhance the background models. Future work will concentrate on the convergence of the stochastic reinforcement learning algorithm for multi-actor cases and applying our approach to other sensory data.

Chapter 7

Performance Improvement by Input Adaptation

This paper focuses on developing self-adapting automatic object detection systems to achieve robust performance. Two general methodologies for performance improvement are first introduced. They are based on optimization of parameters of an algorithm and adaptation of the input to an algorithm. Different modified Hebbian learning rules are used to build adaptive feature extractors which transform the input data into a desired form for a given object detection algorithm. To show its feasibility, input adaptors for object detection are designed and tested using multisensor data including SAR, FLIR, and color images. Test results are presented and discussed in the paper.

7.1 Introduction

Automatic object detection is of great importance for many vision based real-world applications. An automatic object detection system should be able to locate objects of interest in the input images produced by different sensors such as CCD cameras, infrared sensors, radars and multispectral scanners. Although many automatic object detection systems have been developed, their performance is still limited [96]. This paper is motivated by the increased demand for new theories and methodologies to improve system performance [38, 96, 98] and to minimize the effort needed for the development of robust object detection systems. The original contribution of this paper is the idea that the performance of a given algorithm can be improved by adding an adaptor between the input data and the algorithm. This is an input adaptive process and is based on the observation that most

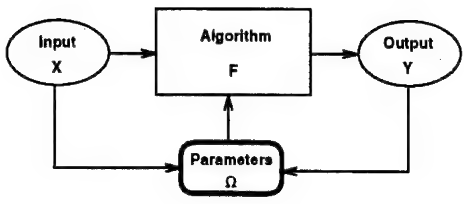


Figure 7.1: Parameter optimizing methodology for performance improvement.

algorithms would perform well if the desired input data can be provided to them. Different kinds of Hebbian-like learning rules are introduced and applied to developing such adaptors. The feasibility of this methodology for performance improvement is demonstrated by experimental results using multisensor data.

7.2 Parameter Optimization Versus Input Adaptation for Performance Improvements

The first methodology is based on the consideration that some algorithms and systems have certain controllability and their performance can be improved by tuning their parameters [12]. To find the best parameter set for the given input data a learning and optimizing process is usually required. This methodology is, therefore, parameter optimizing oriented. As shown in Figure 7.1, parameter optimizing based methodology employs different parameter set for different input data in order to obtain the optimal output. However, this methodology suffers from some inherent shortcomings: (a) It is driven by both the input data and the output data. It has to have an off-line learning phase. This leads to the difficulty of sample collection because some input situations are not predictable. Besides, the off-line learning process is usually time consuming. (b) In order to use the trained algorithm, information about the possible category of the input data is needed before the appropriate parameter set can be switched on. This means that the trained algorithm works only with an additional input identifier which triggers the corresponding parameter set. Certainly the design of such an identifier is as hard as that of the algorithm itself. (c) The performance of an algorithm cannot be always improved by optimizing the parameter set because the gradients of objective functions of some algorithms with respect to their parameters are too small. So not all algorithms can be improved by using this methodology.

The second methodology for performance improvement is based on the observation that most algorithms would perform well if their input data are "friendly", as discussed above.

Thus, the performance of almost all commonly used algorithms can be improved by adding an adaptor between the input data and the algorithm (see Figure 7.2). An ideal adaptor should automatically judge the input data, provide the desired input data to an algorithm, and learn something from this process in order to improve itself in the future.

In comparison with the parameter optimizing based methodology, the input adapting methodology presented in this papaer has some positive features such as: (a) It is suitable for almost all algorithms because the desired input data (not always the perfect input data) always exists for a given algorithm. (b) It is driven only by the input data. So it can work both on-line and off-line. This is very important for real-world and real-time applications. (c) This methodology makes it possible to combine some simple, ready-made available algorithms to build vision systems that exhibit high level performance. Without adding adaptors, these simple algorithms may be unreliable for practical applications, although they have simple structures and are not time consuming.

7.3 Representations Versus Salient Features

Most commonly used algorithms can show good performance only if their input representations have some "friendly" characteristics. To keep their performance high even if the input representations are not so "friendly", adaptors are needed which transform the input representations to some salient features. Thus, an adaptor can also be regarded as a salient feature extractor. The key issue in input adapting methodology for performance improvement is how to design an adaptor or feature extractor for each algorithm at each stage of the representation transformation.

7.3.1 Optimal Feature Extraction

From a mathematical viewpoint, feature extraction is a transformation from a m-dimensional input representation $\mathbf{x}$ to a n-dimensional output representation $\mathbf{v}$, so that $n \leq m$ and for each $v \in \mathbf{v}$ the expected value of $\rho(v)$ is minimized:

$$E(\rho(v)) = \int_{-\infty}^{+\infty} \rho(v)p(v)dv \rightarrow \min, \tag{7.1}$$

where $\rho(\cdot)$ is a "loss" function, $E(\cdot)$ is the risk (the expected value of the loss), and $p(\cdot)$ is the probability density function of v. This means that the transformed representation $\mathbf{v}$ should be less redundant (because of $n \leq m$) and salient (because of $E(\rho(v)) \rightarrow \min, v \in \mathbf{v}$). Thus $E(\cdot)$ is a measure of saliency which depends on the loss function $\rho(\cdot)$.

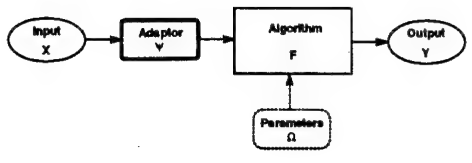


Figure 7.2: Input adapting methodology for performance improvement.

A simple example of the representation transformation is the linear mapping $\mathbf{W}$ which transforms the m-dimensional input representation $\mathbf{x}$ to n-dimensional output representation $\mathbf{v}$ by using

$$\mathbf{v} = \mathbf{W}\mathbf{x} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m)^\top \mathbf{x}. \tag{7.2}$$

In this case, $\mathbf{W}$ is a feature extractor if $\mathbf{v}$ has some nice properties. The feature extractor $\mathbf{W}$ can be realized by using a single-layer linear feed-forward network.

As can be seen, the basic unit of this network is a m to 1 mapping

$$v = \mathbf{w}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{w}, \quad v \in \mathbf{v}. \tag{7.3}$$

The basic unit can also be nonlinear. In this case the m to 1 mapping is formulated by

$$v = \tau(\mathbf{w}^\top \mathbf{x}) = \tau(\mathbf{x}^\top \mathbf{w}), \quad v \in \mathbf{v}, \tag{7.4}$$

where $\tau(\cdot)$ is nonlinear function. The mapping (7.3) or (7.4) is salient or interesting if $E(\rho(v))$ is minimized. The key issue of constructing a feature extractor is thus the design of the loss function.

Before the loss function can be designed, the question of which v is "salient" or "interesting" should be first defined. Certainly, no universal agreement on this question can be expected. Two general definitions about the saliency of v that we have are:

- **Expressiveness:** v is salient if it is *expressive.*
- **Discrimination:** v is salient if it is *discriminating.*

In the following the problem of how to extract these features is addressed.

7.3.2 Expressive Feature

Let us consider a set of m-dimensional input representations $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$ which builds a "cloud" of points in the m-dimensional space. It is clear that each point $\mathbf{x} \in \mathcal{X}$

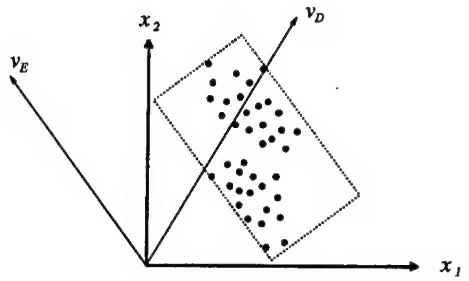


Figure 7.3: A point cloud in a 2-dimensional space.

can be projected onto a direction determined by the vector $\mathbf{w}$ by using Equation (7.3) or (7.4) and the result of this projection is v. Figure 7.3 just shows a case of $m = 2$. Now the problem is which projection direction is interesting.

As shown in Figure 7.3 the first interesting direction is v_E because the projections of all points onto this direction have the maximal variance and v_E is, therefore, *expressive*. It can be proved that v_E is determined by that $\mathbf{w}$ which is the largest eigenvector associated with the largest eigenvalue of the correlation matrix

$$\mathbf{Q} = E(\mathbf{x}\mathbf{x}^\top). \tag{7.5}$$

Let us first define a loss function

$$\rho_G = \frac{1}{2}v^2 \tag{7.6}$$

The risk $E(\rho_G)$ can be calculated by

$$E(\rho_G) = \frac{1}{2}E(v^2) = \frac{1}{2}\mathbf{w}^\top\mathbf{Q}\mathbf{w}. \tag{7.7}$$

Minimizing $E(\rho_G)$ requires

$$\Delta\mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} = \mathbf{0}. \tag{7.8}$$

This leads to famous *plain Hebbian learning* rule

$$\Delta w_i = \eta v x_i, \tag{7.9}$$

where w_i and x_i are the i^{th} component of $\mathbf{w}$ and $\mathbf{x}$ respectively, and η controls the learning rate as usual. It can be seen that Hebbian learning is controlled by both the input (through x_i) and the output (through v).

Equation (7.8) tells that $\mathbf{w}$ is an eigenvector of $\mathbf{Q}$ with eigenvalue 0. But this could never be stable, because $\mathbf{Q}$ necessarily has some positive eigenvalues; any fluctuation having a component along an eigenvector with positive eigenvalue would grow exponentially. It might be suspected that the direction with the largest eigenvalue of $\mathbf{Q}$ would eventually become dominant, so that $\mathbf{w}$ would gradually approach the eigenvector corresponding to the largest eigenvalue with a increasingly huge norm. Certainly, $\mathbf{w}$ does not settle down in any case. There are only unstable fixed points for plain Hebbian learning procedure (7.9).

Let us modify the loss function (7.6) to

$$\rho_E = \frac{1}{2}\left[v^2 - E(v^2)\mathbf{w}^\top \mathbf{w}\right]. \tag{7.10}$$

The risk $E(\rho_E)$ can be calculated by

$$\begin{aligned} E(\rho_E) &= \frac{1}{2}(E(v^2) - E(v^2)\mathbf{w}^\top \mathbf{w}) \\ &= \frac{1}{2}(\mathbf{w}^\top \mathbf{Q}\mathbf{w} - E(v^2)\mathbf{w}^\top \mathbf{w}). \end{aligned} \tag{7.11}$$

Since the risk is continuously differentiable, the optimization of (7.11) can be achieved, via a gradient descent method, with respect to $\mathbf{w}$:

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} - E(v^2)\mathbf{w} = \mathbf{0}. \tag{7.12}$$

Clearly, an equilibrium can be reached if $\mathbf{w}$ is the eigenvector associated with one eigenvalue, say the largest one, of $\mathbf{Q}$ and $E(v^2)$ is just the eigenvalue.

Equation (7.12) leads to the learning rule suggested by Oja [60]. According to this rule, each input $\mathbf{x} \in \mathcal{X}$ is applied to adapt the weight $\mathbf{w}$ by using

$$\Delta w_i = \eta v(x_i - v w_i), \tag{7.13}$$

where w_i and x_i are the i^{th} component of $\mathbf{w}$ and $\mathbf{x}$ respectively, and η controls the learning rate. The learning rule (7.13) is a generalized version of Hebbian learning rule (7.9) which has been so widely applied to develop unsupervised learning networks.

Comparing the loss function (7.10) with (7.6) shows that these Hebbian-like learning rules are based on second order statistics as they usually use second order polynomials for measuring the interest and they lead to extraction of *principal components* (PC) of the input data [48, 60, 78]. It can be proved [48] that minimizing the risk (7.11) is equivalent to maximizing the information content of the output representation in situations where that has a Gaussian distribution.

The direction v_E found in this way allows faithful representation of the input data and the projection of the point cloud onto this direction can also show interesting structure if the cloud contains a few clusters and the separation between clusters is larger than the internal scatter of the clusters. However, the direction v_E can lead us astray if the cloud shows too many isotropically distributed clusters or if there are meaningless variables (x_i's) with a high noise level. In these two cases, the output representation v_E doesn't allow discrimination between clusters (see the example in Figure 7.3). This means that second order polynomials are not sufficient to characterize the important features of an input distribution and all second order statistics based feature extractors cannot provide features which are discriminating enough for recognizing the structure in the input representation.

7.3.3 Discriminating Feature

As shown in Figure 7.3, the second interesting direction is v_D because the projections of all points onto this direction can enable us to better distinguish the interesting structure (clusters) presented in the cloud and v_D is, therefore, *discriminating.*

In order to find this direction, a measure sensitive to distributions which are far from Gaussian is needed. As already discussed, second order polynomials such as shown in (7.10) cannot be used for measuring deviation from normality. To emphasize bi- or multi-modality of the projected distribution, higher order polynomials are required and care should be taken to avoid their over-sensitivity to small number of outliers.

Let us define a loss function

$$\rho_D = v^2 \left[\frac{E(v^2)}{4} - \frac{|v|}{3} \right] = v^2 r(v), \tag{7.14}$$

where $r(v)$ can be regarded as a weighting function. The loss function ρ_D is small if v is close to zero or to $3E(v^2)/4$. Moreover, it remains negative for $v > 3E(v^2)/4$. Thus, ρ_D as an index can exhibit the fact that bimodal distribution is already interesting, and any additional mode should make the distribution even more interesting.

Actually, any radial basis function (see [98]) can be used as the weighting function in Equation (7.14) to design interesting loss functions. The advantage of $r(v)$ used in Equation (7.14) is its connection to the Bienenstock, Cooper, and Munro (BCM) theory of visual cortical plasticity [16, 43].

The expected value of ρ_D is given by:

$$E(\rho_D) = \frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3)$$

$$= \frac{1}{4}E(v^2)\mathbf{w}^\top \mathbf{Q}\mathbf{w} - \frac{1}{3}E(v^3). \tag{7.15}$$

To achieve $E(\rho_D) \rightarrow \min$, the equation

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\left[\frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3)\right] = \mathbf{0} \tag{7.16}$$

should be satisfied. This leads to a learning rule

$$\Delta w_i = \eta \left[v^2 - E(v^2)v\right] x_i, \tag{7.17}$$

where x_i is the i^{th} component of $\mathbf{x}$ and η controls the learning rate.

The difference between the learning rule (7.17) and Hebbian learning rule (7.9) is the fact that the influence of the output on the learning process (the feedback) has been changed from v in (7.9) to $v^2 - E(v^2)v$ in (7.17). This enables the learning rule (7.17) to discover bimodal distributions as Δw_i in (7.17) (unlike in (7.9)) has opposite value depending on if v is larger or smaller than $E(v^2)$.

Unfortunately, the learning rule (7.17) has the same divergence problem like Hebbian learning rule (7.9) and in any case $\mathbf{w}$ does not settle down. One way to prevent the divergence of $\mathbf{w}$ is to constrain the growth of $\mathbf{w}$ by modifying the loss function (7.14):

$$\rho_Z = v^2\left[\frac{E(v^2)}{4} - \frac{|v|}{3}\right] - E(v^2)\mathbf{w}^\top\mathbf{w}. \tag{7.18}$$

This leads to a new learning rule obtained by adding a weight decay to the learning rule (7.17):

$$\Delta w_i = \eta \left[v^2 - E(v^2)v\right](x_i - v w_i). \tag{7.19}$$

So far four different learning rules have been introduced. They are based on four different loss functions (see Table 7.1) and can be applied to extracting expressive and discriminating features. In Table 7.1, EF denotes "expressive features" and DF denotes "discriminating features".

7.4 Adaptive Object Detection

Table 7.1: Different Learning Rules For Feature Extraction

Type	The Loss Function	The Learning Rule	Suitability
I	$\rho_G = \frac{1}{2}v^2$	$\Delta w_i = \eta v x_i$	EF
II	$\rho_E = \frac{1}{2}\left[v^2 - E(v^2)\mathbf{w}^\top \mathbf{w}\right]$	$\Delta w_i = \eta v(x_i - v w_i)$	EF
III	$\rho_D = v^2\left[\frac{E(v^2)}{4} - \frac{\lvert v\rvert}{3}\right]$	$\Delta w_i = \eta\left[v^2 - E(v^2)v\right]x_i$	DF
IV	$\rho_Z = v^2\left[\frac{E(v^2)}{4} - \frac{\lvert v\rvert}{3}\right] - E(v^2)\mathbf{w}^\top \mathbf{w}$	$\Delta w_i = \eta\left[v^2 - E(v^2)v\right](x_i - v w_i)$	DF

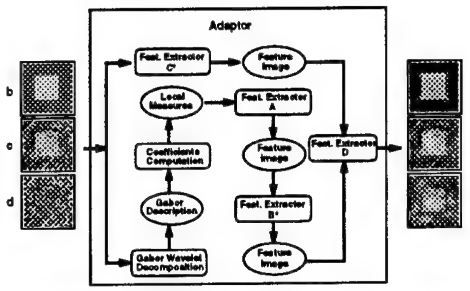


Figure 7.4: Input adapting for the image thresholding algorithm.

7.4.1 Adaptor Design

Figure 7.4 shows an adaptor which is designed for the image thresholding algorithm. The key idea for designing this adaptor is to decompose the input image into some local measure images and then to adaptively extract salient features from these local measure images based on the modified Hebbian learning rules presented above. In order to derive local measures for each pixel in the input image, the quadrature Gabor filter kernels

$$\begin{aligned} G_+(\omega,\phi) &= \exp\left[-\frac{\lambda^2\omega^2(x^2+y^2)}{4\pi}\right] \\ &\quad\cdot\cos[\omega(x\cos\phi + y\sin\phi)] \\ G_-(\omega,\phi) &= \exp\left[-\frac{\lambda^2\omega^2(x^2+y^2)}{4\pi}\right] \end{aligned} \tag{7.20}$$

$$\cdot \sin[\omega(x\cos\phi + y\sin\phi)] \tag{7.21}$$

are applied to decompose the input image $I(x, y)$ by using

$$\begin{aligned} I_+(x, y, \omega, \phi) &= G_+(\omega, \phi) * I(x, y), \\ I_-(x, y, \omega, \phi) &= G_-(\omega, \phi) * I(x, y), \end{aligned} \tag{7.22}$$

where ω and ϕ are the modulation (center) frequency and orientation, respectively, of the Gabor filter kernel; λ is the ratio of the channel bandwidth and the modulation frequency; and $I_+(x, y, \omega, \phi)$ and $I_-(x, y, \omega, \phi)$ are Gabor space image descriptions. From these descriptions it is easy to derive some local measures. In Figure 7.4 the power

$$p(x, y, \omega, \phi) = I_+^2(x, y, \omega, \phi) + I_-^2(x, y, \omega, \phi) \tag{7.23}$$

is used as a local measure of the input image $I(x, y)$. So far m power images can be obtained and m depends on the quantization of ω and ϕ. This means a local measure vector with m elements is associated with each pixel of the input image.

Each element of the local measure vector is a representation to describe the local property of the input image but it is not just the right feature to discriminate clusters depicted in the input image. The most discriminating feature should be found in the m-dimensional local measure space based on the structure presented by all local measure vectors in the input image. This requires a m to 1 feature extractor **A** which is trained by using the learning rule (7.17) or (7.19) as described above.

To reduce high frequency components in the input data two feature extractors **B**$*$ and **C**$*$ are introduced into the adaptor shown in Figure 7.4. They are actually two convolution kernels with $n \times n$ elements which should be trained by using the learning rule (7.9) or (7.13) as described above.

After the convolution using **B**$*$ and **C**$*$ two feature images can be produced which should be integrated by the feature extractor **D** in order to supply desired images for the thresholding algorithm. The feature extractor **D** performs a 2 to 1 transformation and is trained by using the learning rule (7.17) or (7.19).

Now the adaptor is able to produce desired input images for the thresholding algorithm (see Figure 7.4). It is obvious that the output images in Figure 7.4 are better than the input images in Figure 7.4 to be used as the input images for the thresholding algorithm because the object in the center is better discriminated from the background. Thus, the performance of the thresholding algorithm can be improved by using the adaptor.

7.4.2 Experimental Result for Adaptive Target Detection

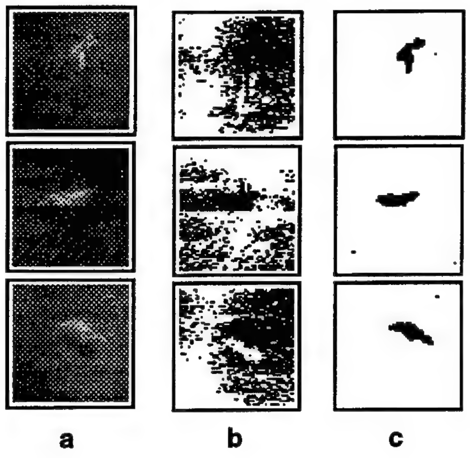


Figure 7.5: Test Result Using SAR Images.

Figure 7.5 shows the test result of target detection system using SAR image data. The column **a** shows the input images. The column **b** shows the test results using the thresholding algorithm. The column **c** shows the test results using the thresholding algorithm plus the input adaptor. It can be seen that even a simple algorithm can perform well if its input data are properly prepared by an input adaptor. This means that adding an input adaptor can enlarge the dynamic range of an algorithm and improve its performance.

Figure 7.6 shows another example of target detection in a FLIR (Forward Looking Infrared) image by using the same system. Again, the image **a** is the input image. The image **b** and **c** show the test results using the thresholding algorithm without and with the input adaptor. As can be seen, the performance of the system is satisfied even when the input image has different properties as used before.

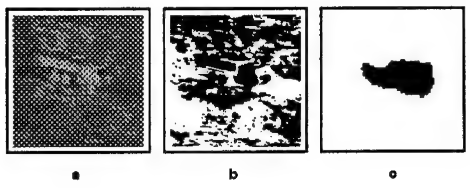


Figure 7.6: Test Result Using a FLIR Image.

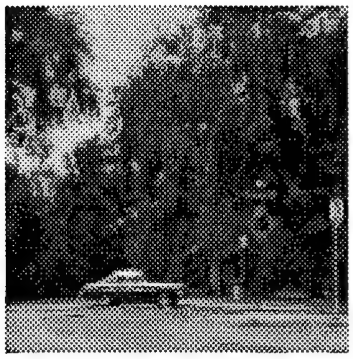


Figure 7.7: An outdoor scene with a car and a yellow traffic sign near the car.

7.4.3 Detection of Colored Objects

The first step to design a system for object detection from a color image is the specification of an object. An object, for instance, can be defined as a connected region in a color image which has a special shape such as circle or rectangle. It can also be defined as a connected region which has a given color topology such as a red region surrounded by a yellow region. In this paper, an object in a color image is defined as a connected region which is small and well colored. Figure 7.7 shows a sample image. The scene is photographed approximately every 15 minutes over a four hour period by using a fixed JVC GXF700U color video camera. A total of 20 image frames are obtained in this way and only four of them are selected for the experiment. The time and the weather condition of these four color images are: Frame 1, 1:20pm, Sunny; Frame 5, 2:15pm, Sunny; Frame 9, 3:15pm, Sunny; and Frame 13, 4:45pm, Sunny.

The colors of the car and traffic sign in Frame 13 are subdued since they are located under the shadow of the trees when Frame 13 was taken. However, these objects are well colored in Frame 1, because there was no shadow at 1:20 pm when Frame 1 was taken. This can be seen if all the pixels of both Frames are mapped into the RGB color space (see Figure 7.8). The R, G, and B values of all pixels in Figure 7.8 are normalized in the range $[-0.5, 0.5]$. As shown in Figure 7.8, all pixels in Frame 13 are located along the line segment between the point $[-0.5, -0.5, -0.5]$ and the point $[0.5, 0.5, 0.5]$. This line can be thought of as a colorless line. This means that the saturation or the color of all pixels in Frame 13 is relatively low because they are located close to the colorless line.

On the contrary, some pixels in Frame 1 are located away from the colorless line and the

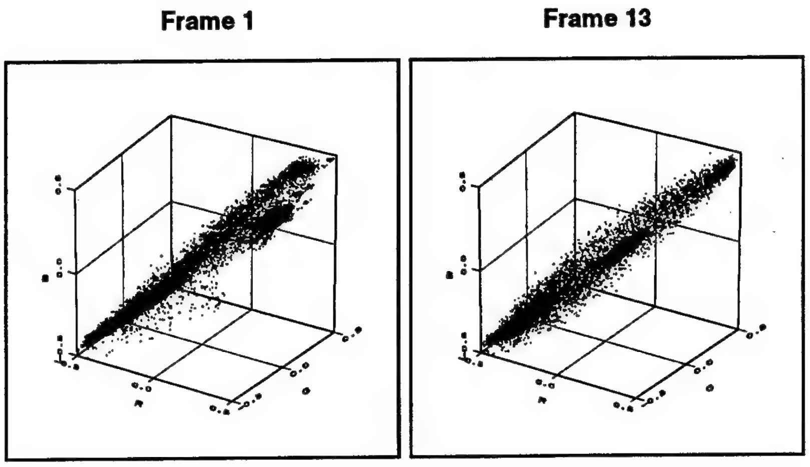


Figure 7.8: Image pixels are mapped into the RGB color space.

saturation of these pixels is relatively good. These pixels are well separated from the pixel group around the colorless line and can be regarded as outliers of this pixel group. Thus, they can be defined as well colored pixels which build regions of interest in Frame 1. In the following we describe how to develop an adaptive system to find these outlier pixels.

It is first interesting to know what happens if all pixels in a color image are applied to train the 3 to 1 feedforward network by using Hebbian-like learning rules shown in Table 7.1. The weights of this 3 to 1 feedforward network obtained after training by using the learning rule II and III in Table 7.1 are shown in Table 7.2 and Table 7.3, respectively. The trained network can then be used to map a color image into a gray scale image. Figure 7.9 shows the gray scale images mapped for Frame 1 and Frame 13. The first row in Figure 7.9 shows the images mapped by using the weights listed in Table 7.2, while the second row in Figure 7.9 shows the images mapped by using the weights listed in Table 7.3.

If the two mapped images shown in the second row of Figure 7.9 are compared, we can see that the car and the traffic sign are much better separable from the background in Frame 1 than in Frame 13. This means that, although Frame 13 has three color channels R, G and B, the color information encoded in this frame is so weak that this frame can be regarded as almost colorless. In fact, most of the information in Frame 13 is encoded in a gray scale image which is the right image in the first row of Figure 7.9, because it was obtained by using the most expressive mapping and this mapping was determined by using all the pixels in Frame 13 as the training data and the modified Hebbian learning rule II in

Table 7.2: Weights after training by using the learning rule II.

Image Used for Training	w_1	w_2	w_3
1	0.576884	0.592581	0.562207
5	0.585867	0.590048	0.555533
9	0.583240	0.567947	0.580755
13	0.585854	0.577843	0.568225

Table 7.3: Weights after training by using the learning rule III.

Image Used for Training	w_1	w_2	w_3
1	0.691818	-0.716642	0.088382
5	0.685157	-0.724709	0.073194
9	0.727544	-0.685147	-0.035408
13	0.657560	-0.749287	0.078640

Table 7.1.

The well colored objects such as the car and the traffic sign in this image (the left image of second row shown in Figure 7.9) are separable from the background. To understand this, all selected frames are transformed by using such mapping (see the first row of Figure 7.10) and compared with their saturation (see the second row of Figure 7.10). It is clear that the adaptive mapping obtained by using the modified Hebbian learning rule III or IV in Table 7.1 discriminates well-colored objects from the background. The left image of Figure 7.11 shows the object detection result from Frame 1 after thresholding the results shown in the top left image of Figure 7.10. This image can be further used for post-processing. The right image of Figure 7.11 shows the post-processing result using morphological filtering followed

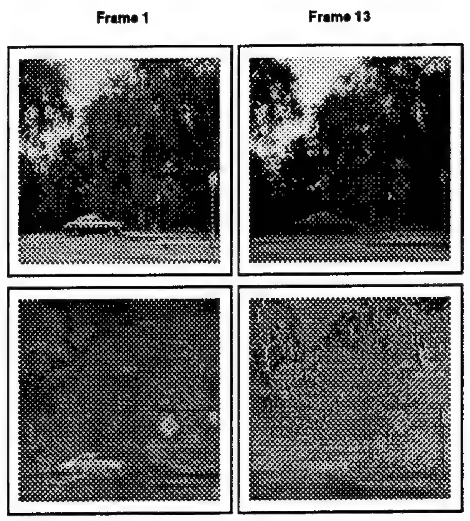


Figure 7.9: Grey scale images obtained by adaptive mapping.

by color based filtering.

7.5 Conclusions

In this paper, the attention was paid on how to to improve the performance of object detection systems by adding the adaptability to ready-made available algorithms without changing their internal structure. The input adapting based approach presented here provides a promising solution to improve the performance of pattern recognition and computer vision algorithms and systems to meet requirements of real-world applications.

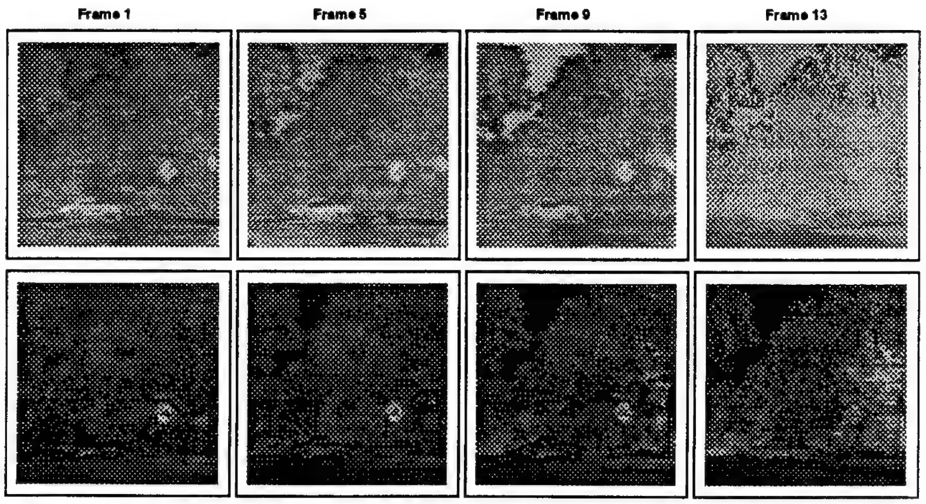


Figure 7.10: The discriminating mapping used to color images and compared with the saturation mapping.

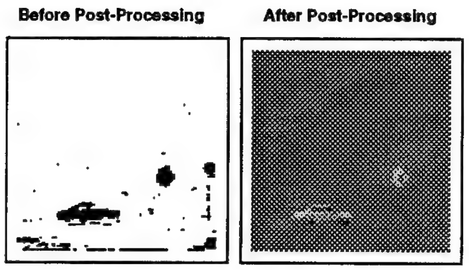


Figure 7.11: The object detection from Frame 1, before and after post-processing.

Chapter 8

Case-Based Learning of Recognition Strategies

8.1 Introduction

Photointerpretation (PI) has been an important application domain of image understanding (IU) techniques for about two decades. An important goal of PI or image exploitation (extraction of intelligence from image data, particularly aerial imagery) is to aid reconnaissance tasks, such as airfield, port, and troop movement monitoring. The problem of PI is one of identifying instances of "known" object models in images acquired from a platform, such as by a satellite or a reconnaissance aircraft. Like PI, automatic target recognition (ATR) is also concerned with finding instances of known targets in the input sensor data. Model-based object recognition is a challenging task under real-world conditions such as occlusion, shadow, cloud cover, haze, seasonal variations, clutter, and various other forms of image degradation. Additionally, ATR scenarios are characterized by multi-modal imagery, low resolution, and camouflage. All of these problems put heavy requirements on any IU system to be robust.

Automatic acquisition of recognition strategies in dynamic situations has been a bottleneck in the development of automated IU systems applied to real-world problems, such as PI and ATR. The problem occurs while matching a stored object model to an input instance of that model and is attributed to the initially unknown pose of object and the varying environmental conditions. During the process of image/scene understanding, a human relies heavily on the memory of past cases and experience. We use the Case-Based Reasoning (CBR) paradigm in which "past" experiences are stored in memory as cases and are used to solve a new problem case. Similar cases can be combined to create problem solving shortcuts or to anticipate problems in new situations. The set of cases is prioritized and a strategy for the current problem is generated and executed. Various combinations of cases are created until a successful solution is reached.

8.2 Learning Recognition Strategies

Figure 8.1 describes our approach to learning recognition strategies for real-world object recognition tasks. The main learning paradigm employed in our recognition scheme is Case-Based Reasoning. The detailed CBR-based recognition framework shown in Figure 8.1 consists of four subtasks: (a) the generation of goal-directed recognition strategies using CBR, (b) the construction and maintenance of the Generalized Case Library (GCL) that collects past situations and corresponding actions, (c) the development of efficient algorithms for matching new situations to previous cases, and (d) the generalization of new cases using a variation of Explanation-Based Learning (EBL). Additionally, our approach also addresses the problems of indexing into the object model data base and the verification of object hypotheses. This latter task consists of two main parts: (a) the creation and refinement of the decision structures for indexing, using a variant of the Conceptual Clustering (CC) learning technique, and (b) the implementation of the indexing and matching algorithms. In this report, we focus on the CBR-based framework.

8.2.1 Case-Based Reasoning (CBR)

Case-based approaches are characterized by how the learner represents what it has learned so far, as well as the analogical methods which are used to transfer the learned experience. Human expertise in problem solving is largely dependent on past experiences. This idea has influenced the evolution of Case-Based Reasoning [5, 45, 73]. A related approach is that of reasoning by analogy [4, 28]. In CBR, "past" experiences are stored in memory as cases and are used to solve a new problem case. Given a problem to be solved, the case-based method *retrieves* from the memory the solution to a similar problem encountered in the past, *adapts* the previous solution to the current problem, and *stores* the new problem-solution packet as another case in the memory.

There are several advantages of CBR as a learning paradigm. First, CBR has the capability of anticipating and therefore avoiding past mistakes as well as focusing on the most important aspects of a problem first. All of these lead to an increase in efficiency over time. Second, the learning process is fairly uncomplicated, since CBR does not require causal models like inductive learning or extensive domain knowledge like analytic learning. Third,

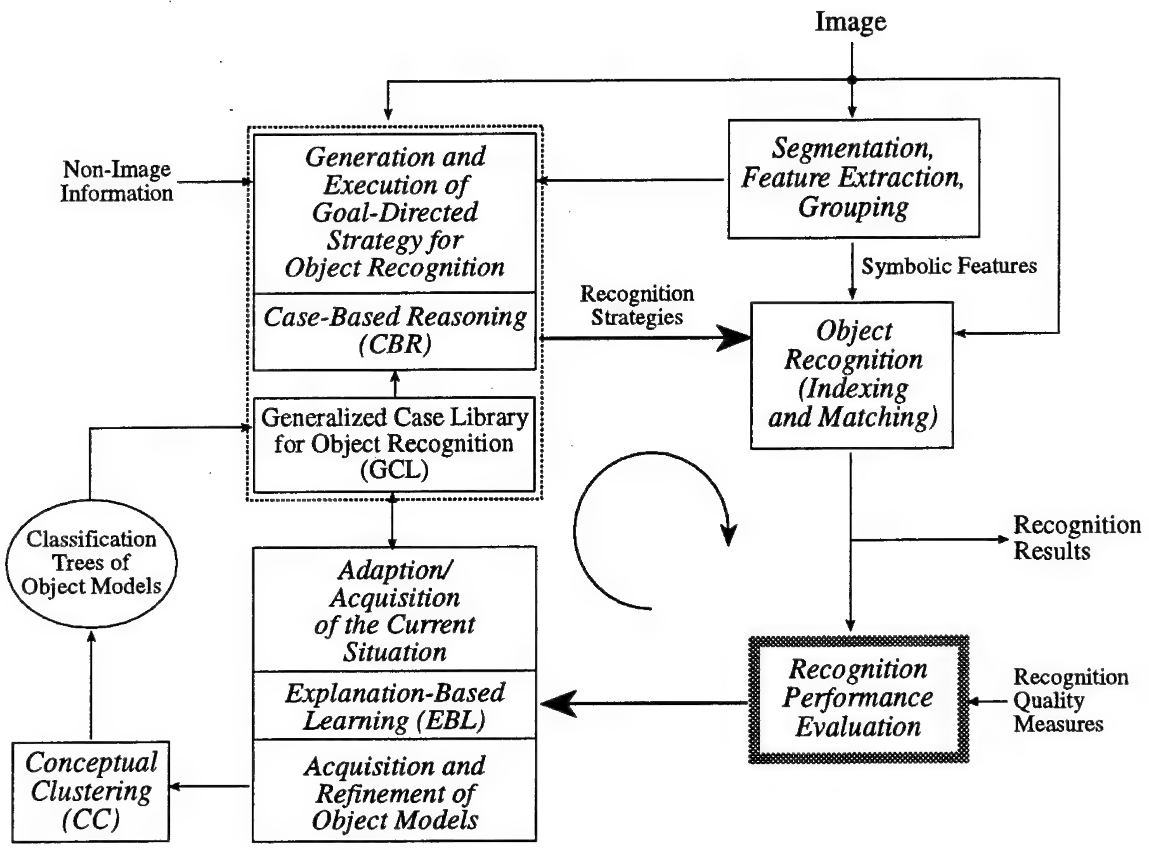


Image
Non-Image Information
Generation and Execution of Goal-Directed Strategy for Object Recognition
Case-Based Reasoning (CBR)
Generalized Case Library for Object Recognition (GCL)
Segmentation, Feature Extraction, Grouping
Symbolic Features
Recognition Strategies
Object Recognition (Indexing and Matching)
Recognition Results
Classification Trees of Object Models
Adaption/ Acquisition of the Current Situation
Explanation-Based Learning (EBL)
Acquisition and Refinement of Object Models
Conceptual Clustering (CC)
Recognition Performance Evaluation
Recognition Quality Measures

Figure 8.1: A CBR framework for learning recognition strategies. EBL generalizes cases and along with CC it facilitates automatic knowledge acquisition of object models.

the individual or generalized cases can also serve as explanations. Fourth, the process is scalable. Fifth, the knowledge acquisition bottleneck is relatively simple to solve in CBR than in conventional learning systems. This is because individual cases interact a little among themselves unlike the rules. The major concerns with CBR are the selection of the indexing scheme to organize cases in the memory, the method for choosing the most relevant cases at reasoning time, and the adaptation heuristics to modify previous cases to fit the current problem.

There are two major types of case-based approaches: *interpretive/classification* (or precedent-based) CBR, and *problem solving* CBR. In the precedent-based CBR, the task is to decide whether or not a new case should be treated like one of the stored cases based on similarities and differences between the two. This is done by generating a pro's and con's analysis from a comparison of the two cases. In problem solving CBR, a solution for the new problem is formulated by suitably modifying past solutions. In either approach, a proposed solution must be verified for appropriateness. This is particularly important if the derived solution is based on "unexplained" experiences. This verification process is akin to an evaluation procedure associated with any learning process. An interpretive CBR is used in such evaluation process to provide a check on the use of knowledge derived from experience.

8.2.2 CBR in IU

Current model-based IU approaches to object recognition generally utilize only the geometric descriptions of object models, i.e., they emphasize the recognition problem as a characteristic of individual object models only. However, there are various factors, such as contextual information, sensor type, target type, scene models, and related non-image information that may influence the outcome of recognition in real-world applications such as ATR, PI, navigation. Humans also rely on such ancillary information for object recognition and scene understanding. For example, it is well known in the intelligence community that oxen yoked to water pumps in Southeast Asia resemble anti-aircraft artillery in aerial images [2]. Thus, without the knowledge of the area being examined, an image analyst or an automated PI system may be misled easily. Thus, prior experience in addition to object/sensor models is important for devising efficient and robust recognition strategies to deal with noisy data or occluded targets against complex backgrounds.

Prototypical situations (cases) observed in the past are useful for the recognition of objects as well as for the assessment of entire scenes. An example of a case in the PI context is given in Figure 8.2. Each path from the root node to a leaf node in the tree represents a single case. The path incorporates the information normally used at each level in an object recognition task, e.g., aircraft recognition. It includes contextual information, e.g., airfield, scene type, e.g., tarmac parking areas, the best object recognition strategy, e.g., selection of segmentation, feature extraction, recognition algorithms and their parameters, and corresponding image analysis goals, e.g., finding instances of transport aircraft such as *Hercules*. A case of ATR would additionally include sensor type, terrain, and radiometric information.

Case-based methods are best suited to problems for which *many* training cases are available, perhaps with many *exceptional* cases, and it is *difficult* to specify appropriate behavior using abstract rules. Most IU applications, such as ATR and PI, are characterized by large-volume image exploitation corresponding to a variety of scenarios, many of which re-

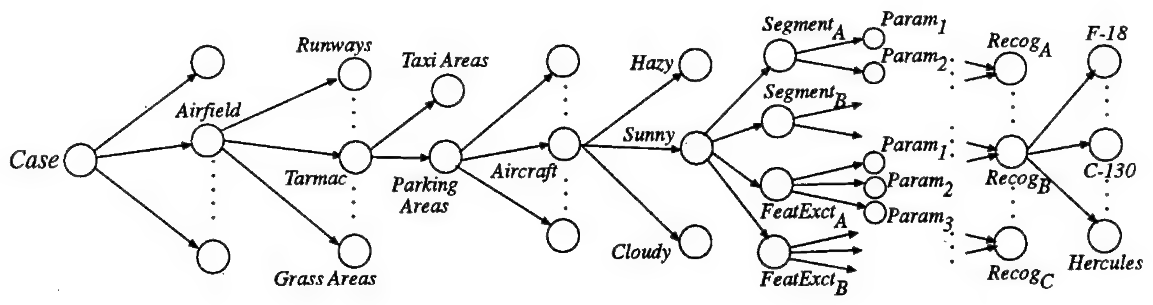


Figure 8.2: Representation of a case in the photointerpretation context.

quire unique analysis. Besides, IU for unstructured environments is difficult to formalize in terms of rules that are general enough to be applicable to diverse situations. For example, recognition of a *Hercules* aircraft in a parked area of the tarmac under sunny condition has been successful in the past by following the path from the root node to the leaf marked "hercules" in the case representation of Figure 8.2. However, the same path may not lead to a successful recognition of an F-18 aircraft. Thus, the *case* of recognizing a *Hercules* is not the same as that of an F-18.

8.2.3 Learning Method

The learning approach is concerned with (a) building new cases, (b) generalizing and refining existing cases, for a particular application. As indicated in Figure 8.1, the relevant knowledge is accumulated in the generalized case library. For updating and indexing into the GCL we use a combination of two different learning strategies: CBR is used primarily for retrieving the relevant earlier experiences and updating (restructuring) the knowledge base; CC is used for maintaining decision structures (classification trees) that allow efficient object recognition at run time.

The GCL is the collection of knowledge that allows the system to perform object recognition and scene assessment. It is a dynamic body of information that represents the experience base of the object recognition system. For efficient indexing, the GCL is represented as a structured hierarchy of individual cases. Each case, in turn, is represented using scripts and memory organization packets (MOPs) which are meta-scripts [80, 81]. These data structures are appropriate for episodic memory or time sequences of episodes which are equivalent to the sequences of computational steps/recognition strategies in our case.

Since scripts contain more specialized information, these are used for lower-levels of a case structure. The MOPs allow representation of more generic knowledge such as an airfield which can be instantiated and specified for recognition of multiple aircraft types.

When a new problem situation or IU task is encountered, e.g., recognition of aircraft on tarmacs, the process of interpreting and assimilating the new task in CBR framework breaks down into the following steps:

- *Assign Indices* – Features of the new task are assigned as indices characterizing the task. For example, "tarmac" and "aircraft" can be used to characterize the task as "aircraft-on-tarmac" which will be a particular subtask of "aircraft-in-airfield" task.
- *Retrieve* – The indices are used to retrieve from memory a similar case encountered in the past based on similarities and differences. The past case contains the prior solution. For example, a case which has involved aircraft on tarmac instead of grass areas.
- *Modify* – The previous solution is adapted to the current task, resulting in a proposed solution. For example, the previous recognition may have occurred under sunny conditions which required detection of shadows, while the weather condition for the current task is cloudy. Thus, the previous case is modified by eliminating all computational steps involving shadows.
- *Test* – The proposed solution is carried out. It may lead to success or failure. For example, the parameters of the segmentation algorithm for detecting regions of interest may have been retained as the same as in the previous case. On the other hand, the contrast of the current image may be low due to cloudy weather condition, thereby, requiring somewhat different segmentation parameter set.
- *Assign and Store* – If the solution succeeds, then indices are assigned to it and the solution is stored as a working solution. The successful plan is then incorporated into the case memory. If the solution is not too different from the proposed solution, then it affects the script of the existing case a little.
- *Explain, Repair, and Test* – If the solution fails, then the failure is explained, the working solution is repaired, and the test is again carried out. The explanation process identifies the source of the problem. For example, new segmentation parameters are selected when recognizing aircraft under cloudy weather condition. The predictive features of the problem are incorporated into the indexing rules to anticipate this problem in the future. For example, "aircraft-on-tarmac" index is extended to "aircraft-on-tarmac-sunny" and "aircraft-on-tarmac-cloudy". The failed plan is repaired to fix the problem, and the revised solution is then tested. The rest of the plan

is carried out with new segmentation parameters in our example. A new case is then created in the memory to handle this new situation.

The results of the CBR-generated strategy are passed to the interpretation and evaluation component. Case indexing and matching is performed using the intermediate visual concepts. The different recognition states are: complete recognition, incomplete recognition, object occlusion, object model acquisition, object model refinement, and recognition failure. Now, three situations may arise. First, if the strategy is very similar to one of the cases extracted from the GCL, no learning takes place. In this instance, the system has encountered an "ordinary" image interpretation task in which the current collection of system knowledge is adequate. Second, if the strategy is an extension of an existing case (i.e., the existing case represents a subset of elements of the new strategy), a case refinement operation may be necessary. The new strategy and its associated case are sent to the EBL module to determine if any new information should be included in the existing case. Third, if a unique combination of existing cases has been utilized to create a novel strategy for a given problem, a case acquisition operation is required. The new strategy is passed to the EBL, which applies its system control knowledge in order to remove irrelevant details and conceptualize the scope of the strategy. This new strategy is then inserted as a new case into the GCL. The CBR and the EBL paradigms are combined in a complementary manner. CBR has the ability to index into a large number of potential solutions and select a set of cases that match the characteristics of the current object recognition task. However, the performance of CBR degrades with the size of the case library and also by the amount of irrelevant detail retained in the stored cases. EBL compensates for this by learning only the concepts underlying the individual cases before adding the conceptual abstraction of the cases to the GCL. On the other hand, since CBR combines a set of previous cases to create a single new case for the current problem, any bias of the EBL component towards a particular training example will be greatly reduced. In summary, CBR allows the capture of context and domain-specific information to improve recognition performance over time.

8.2.4 An Example

An example that illustrates the use of CBR for high-level object recognition is given in Figures 8.3–8.4. A knowledge-based technique initially identifies several regions of interest (ROIs) in the image that are likely to contain aircraft. One such ROI and its corresponding segmentation results are shown in Figures 8.3(b) and 8.3(c), respectively. Also shown in Figure 8.3(c) are the dominant axes of an aircraft structure along the wing and the fuselage. (The third axis corresponding to the shadow of the wing is found to be part of a shadow region and is removed subsequently.) The most "salient" features (with regard to edge strength and global connectivity) and the identified shadow lines are shown in Figures

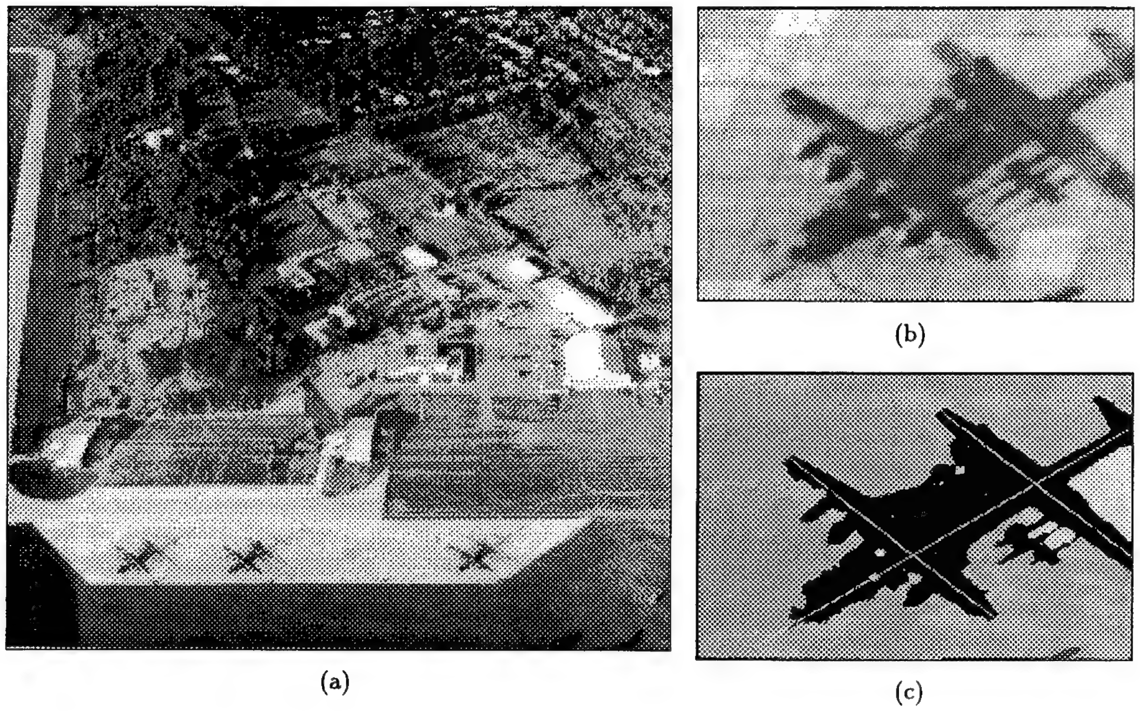


Figure 8.3: High-level object recognition based on CBR. (a) Original image; (b) Initial region of interest (ROI); (c) Extracted dominant axes.

8.4(a) and 8.4(b), respectively. Notice that most of the front edges on both wings are missing from the extracted line group.

A composite structure detection step identifies trapezoid-like shapes that are characteristic of wings, tails, and rudder in non-shadow lines (Figure 8.4(b)). Next, an evidence-based dynamic reasoning process seeks to instantiate one of these composite structures (that are aligned with the dominant axes) as a wing. This situation is shown in Figure 8.4(c). The support for this hypothesis, however, is weak, as there is no evidence for the other wing (i.e., no trapezoid-like structure was detected that is aligned with the same dominant axis). Subsequently, less "salient" line features are acquired (Figure 8.4(e)) and a trapezoid-like structure is detected by relaxing the thresholds of the perceptual grouping process. The final recognition result is shown in Figure 8.4(f).

The experiences gained in this recognition "case" are:

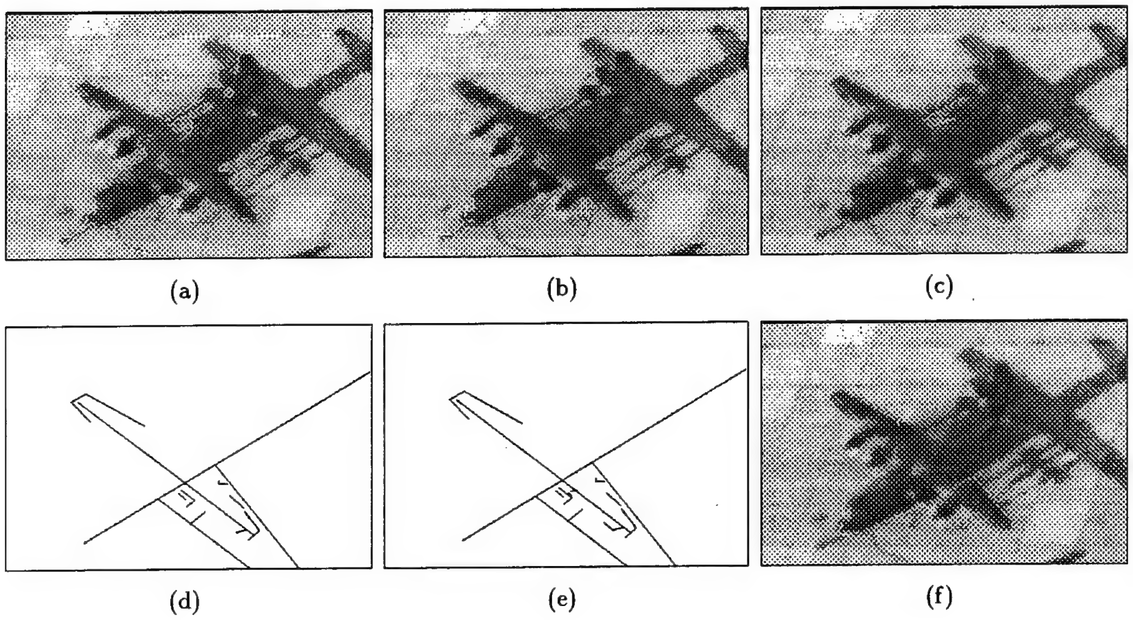


Figure 8.4: High-level object recognition based on CBR *(continued).* (a) Fitted straight lines; (b) Detected shadow lines; (c) Trapezoid shapes in non-shadow groups; (d) Hypothesized right wing and projected left wing; (e) Emergence of additional non-shadow lines; (f) Final recognition result.

- Shadow and object regions are similar (Figures 8.3(a)–(a)), therefore the rear part of the aircraft could not be recovered (Figure 8.4(f)) without using sensor/platform information.
- Relative positions of the sun and the sensor had given rise to specularity along the leading edges of the wings, making these hard to detect from edge information (Figures 8.4(a) and 8.4(d)).
- Evidence of engines had been helpful in hypothesizing a wing (Figure 8.4(d)).

Additional information in this case includes the sun angle, sensor position, sensor/platform parameters, segmentation parameters, directions of shadow regions in a ROI, etc. Clearly, such a "case" is valuable when the task is to investigate another ROI, say the one next to the current one in Figure 8.3(a) which contains another aircraft of the same type (i.e., a *Hercules*). The recognition algorithm will use the same segmentation parameters, will try to verify the front parts of the airplane first, and will know that the leading edges of the wings may be difficult to detect.

8.2.5 Implementation Issues and Performance Evaluation

There are several issues of practical importance in implementing a CBR-based recognition system. These issues are,

- representation and contents of a case in the memory,
- memory organization and selection of indexing rules and search algorithms,
- incorporation of changes over time in the cases and the indexing rules,
- recognition of a new situation as similar to a previous case, i.e., the choice of similarity metrics,
- adaptation of old solutions to new problems, i.e., selection of modification rules,
- acceptance or rejection of a new case that is in conflict with a previous case, i.e., explaining the differences between two problem situations,
- learning from mistakes and devising the repairing rules.

Unlike the rule-based systems, the rules for indexing, modification, and repair do not make up the principal knowledge base but, rather, independent support modules. Thus, the complexity involved is less severe than in most rule-based systems. However, the theory of case-based reasoning suggests that these rules would themselves be acquired by experience from cases through a recursive application of the CBR algorithm. That is, the system would derive rules for indexing, modification, and repair from cases and experience.

The evaluation of the performance of a CBR system can be quite complex due to the nature of the represented knowledge. One way to express the recognition success would be to note the similarity between two problem situations. If these situations are identical, then one would expect identical recognition results. The performance difference would increase with the difference in the situations. Finding a single difference (or similarity) metric would be quite complex as there may exist a number of alternatives to compare two situations. Thus, a multi-objective criterion function would be more appropriate. One could simply focus on the various rules for indexing, modification, and repair to evaluate the performance of a CBR system. For example, the hit vs. miss ratio in retrieving cases from the memory using the indexing rules can be one measure. Various tools from the field of memory management can be used as potential measures to evaluate the efficiency of memory management in a CBR system, e.g., memory usage, memory fragmentation, distributed vs. centralized memory, dynamic memory organization.

8.3 Future Work

Our initial goal of learning recognition strategies using case-based approaches would be limited to PI applications. We have already developed an aircraft recognition system for this purpose and are in the process of extending it further. Currently, this system can handle quite complex imagery and the variabilities present in such images would be ideal for a case-based approach. We have presented some results using this system in this report and sketched our case-based approach. Since our focus is on developing recognition strategies through a learning process, we are minimizing our effort to design appropriate CBR tools. Thus, we are investigating a number of such tools that are currently available commercially, e.g., ESTEEM, CBR-Express, REMIND, and through educational institutions, e.g., Mem -1, Tub-Janos. Some of these allow user-defined similarity measures and also some limited amount of induction. Nonetheless, these tools should provide scope for some initial experimentation, although in the long run these would not suffice since they are developed for non-IU applications. Our future effort would also be directed towards detecting and recognizing other kinds of targets besides aircraft and we would also like to explore ATR applications of our CBR-based learning system.

Chapter 9

Learning Composite Visual Concepts

9.1 Introduction

The context of the learning problem addressed here is *structural object recognition*, which is based on the assumption that structural primitives, extracted from the image in a bottom-up fashion, can be used to describe and recognize the objects of interest. The main advantage of this approach is that it facilitates (at least in principle) recognition under object and aspect variations and, as a recognition-by-components approach, under partial occlusion.

The main problems associated with the structural recognition approach are (a) the computational expense for matching structural object descriptions, (b) the reliable extraction of structural primitives from the image, and (c) the descriptive limitations of the commonly used structural features. The combinatorial problems associated with matching structural descriptions call for methods to limit the search space. When object models are complex, their direct instantiation, either in a top down or a bottom-up, becomes impractical. A logical solution is to describe objects as assemblies of smaller substructures (intermediate visual concepts) that can be instantiated with much less effort. Perceptual grouping methods (e.g., [49, 72, 79]) make use of this fact by using simple geometrical relationships (e.g., collinearity, cotermination, parallelism, etc.) to assemble primitives into more complex features. However, due to the domain-independent specification of perceptual groupings, their "indexing power" is insufficient in applications with more than a few object categories. Another weakness of current structural recognition techniques is their reliance upon a single type of primitive feature, which leads to low redundancy and inappropriate descriptions.

We address the first problem by *learning* significant composite structures that are hierarchically assembled from geometric primitives and serve the purpose of intermediate goals for partial recognition. The other two problems are approached by using a larger variety of *different* structural feature types and corresponding object representations, thus achieving a higher level of redundancy. For the recognition framework we adopt a model-based hypothesize-and-test approach that consists of three main steps: primitive extraction, model-base indexing, and model verification. These three steps operate in a *bootstrap fashion*, i.e., the process starts in a bottom-up mode by extracting primitives and combining them in a meaningful way up to a point when a plausible object hypothesis can be made. Then the recognition process turns into a goal- (model-) directed search and verification process.

The bottom-up part of the recognition process can be viewed as a multi-stage grouping process. At the lowest level, individual pixels are grouped to form the structural primitives, e.g., straight line segments, arcs, regions, etc. At the intermediate-level, the structural primitives produced by feature extraction are combined into more complex structural arrangements, usually biased by perceptual (i.e., domain-independent) constraints. The main goals of the second grouping step[1] are to

1. combine structural features in a way that they are likely to belong to the *same* object, thus reducing the number of "clutter" features that have no correspondence in the model structure and

2. to produce more expressive, object-specific entities that allow effective indexing into the model base.

It is the second item that is our main focus in this part of the project. We need to ask the question, which properties, apart from being perceptually significant, should be incorporated into the grouping process. We believe that, in order to lead to useful object indices, this second set of grouping criteria cannot be model- or domain-independent but needs to be adjusted to the particular application domain, the objects involved, and the context in which they appear. The value of a particular feature group depends mainly upon (a) its *indexing power*, i.e., its capability to select a specific object (or a small set of objects) and (b) its *operationality*, i.e., the effort needed to instantiate it. The general approach for the use of learning to come up with the most effective feature groupings is described in the following.

[1]This step is the one commonly referred to as "grouping."

9.2 General Idea

The intermediate-level part of the project is focused on the problem of "inventing" new composite structural features (intermediate visual concepts) to improve recognition performance. We use intermediate visual concepts that are directly related to the application domain. For this purpose, we select certain high-order assemblies of primitive features which are both perceptually salient and sufficiently distinct to allow very efficient indexing. We employ a two-step grouping strategy that consists of

1. a domain-independent perceptual grouping stage (which ensures perceptual saliency of the selected groups to cope with over-segmentation), followed by

2. a model-based grouping process that is domain-dependent. The high-order, model-based groups are formed as assemblies from the lower-order perceptual groups.

Current perceptual grouping methods (e.g., [49, 72, 79]) are based on (a) a single type of primitives and (b) grouping rules that are predetermined and not adapted to the application domain. The use of a single feature type has the advantage of simple representations and grouping criteria that can be evaluated efficiently. Also, the corresponding structural descriptions are independent of the problem domain. The disadvantages are that

1. the perceptual "saliency" of groupings between different types of primitive features is not used,

2. groupings based on a single feature type are inherently brittle, and

3. fixed, domain-independent grouping rules are not suitable for dynamically changing scenes.

In our approach, we combine multiple types of structural features at the intermediate level, such as line segments, conic sections, corners, inflection points, blobs, etc., in order to increase the descriptive power and robustness (through higher redundancy) of the "polymorphic" feature groupings. The problem of grouping polymorphic features is more challenging than grouping features of the same kind, with regard to the representations and grouping algorithms involved.

The selection and generalization of the intermediate visual concepts is critical in order to in-sure optimal recognition performance. It requires knowledge of the application domain, the imaging process, the behavior of the perceptual grouping stage, and the recognition utility of the intermediate visual concepts. We use Explanation-Based Learning (EBL) to solve this special knowledge acquisition problem. EBL is useful in this context to detect inherent pattern regularities and to generalize patterns, i.e., to determine the simplest description with respect to a given set of operators. In summary, the strategy at this level involves:

1. The use of a two-stage grouping strategy that involves (a) perceptual grouping and (b) model-based grouping with a database of generalized visual concepts.
2. The use of EBL to automatically infer the most useful intermediate visual concepts by applying the entire recognition "engine" to real examples.
3. The use of "polymorphic" feature groupings based on multiple feature types.

The main advantages we expect from this strategy are:

1. A significant reduction of the overall search complexity for structural model instantiation by using high-order intermediate visual concepts.
2. Increased robustness and indexing power from the use of polymorphic groupings.
3. Adaptation of grouping processes to application domains and environmental conditions.

9.2.1 Example

In the aircraft picture shown in Figure 9.1 it is evident that the groups of lines that compose the wings, tails, and rudders, form high-order groupings that are characteristic for many types of aircraft. Obtaining a conceptual description of certain configurations, e.g., the trapezoid that forms the wings, is useful for improving the recognition of other aircraft.

9.2.2 Goals

The main goals at the intermediate level are to automatically acquire new visual concepts from examples, using Explanation-Based Learning and incorporating polymorphic feature groupings. We shall demonstrate that the use of domain- and object-specific grouping, in combination with traditional perceptual grouping, can significantly improve the efficiency of indexing and object recognition.

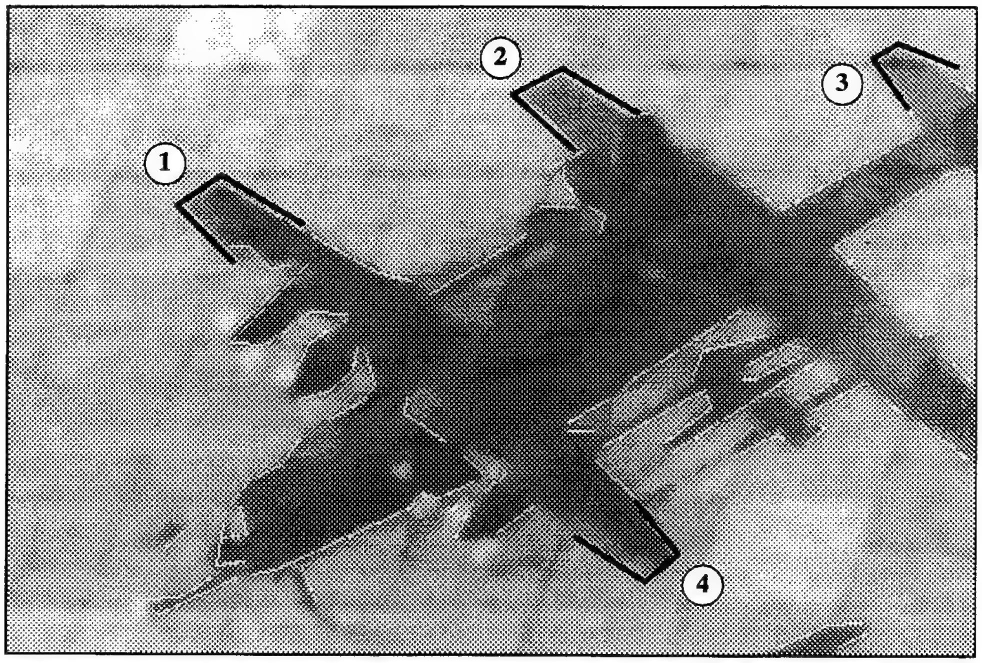


Figure 9.1: Domain-specific, composite visual concepts are formed by combining perceptually salient low-order groupings. Here only straight line segments are used as initial primitives. An example for a simple intermediate-level concept is the typical trapezoid shape found at the ends of the aircraft wings. Four instances (1–4) of this concept are outlined and marked in this image.

9.3 Approach

The instantiation of visual concepts is performed in a two-stage process (Figure 9.2). Initially, the simple features extracted from the input image by various different selection mechanisms (e.g., straight line segments, conic segments, homogeneous blobs, etc.) are grouped using domain-independent perceptual grouping criteria. Examples for the grouping criteria are collinearity, cotermination, parallelism, proximity, relative size, symmetry.

At the second stage, domain-specific models of high-order composite structures (intermediate visual concepts) that have been found useful for recognizing objects guide the grouping process. Visual concepts are learned by the system (see below) and stored in a local database that is continually updated. Only those groupings are considered here that were found perceptually significant at the initial perceptual grouping stage. During actual

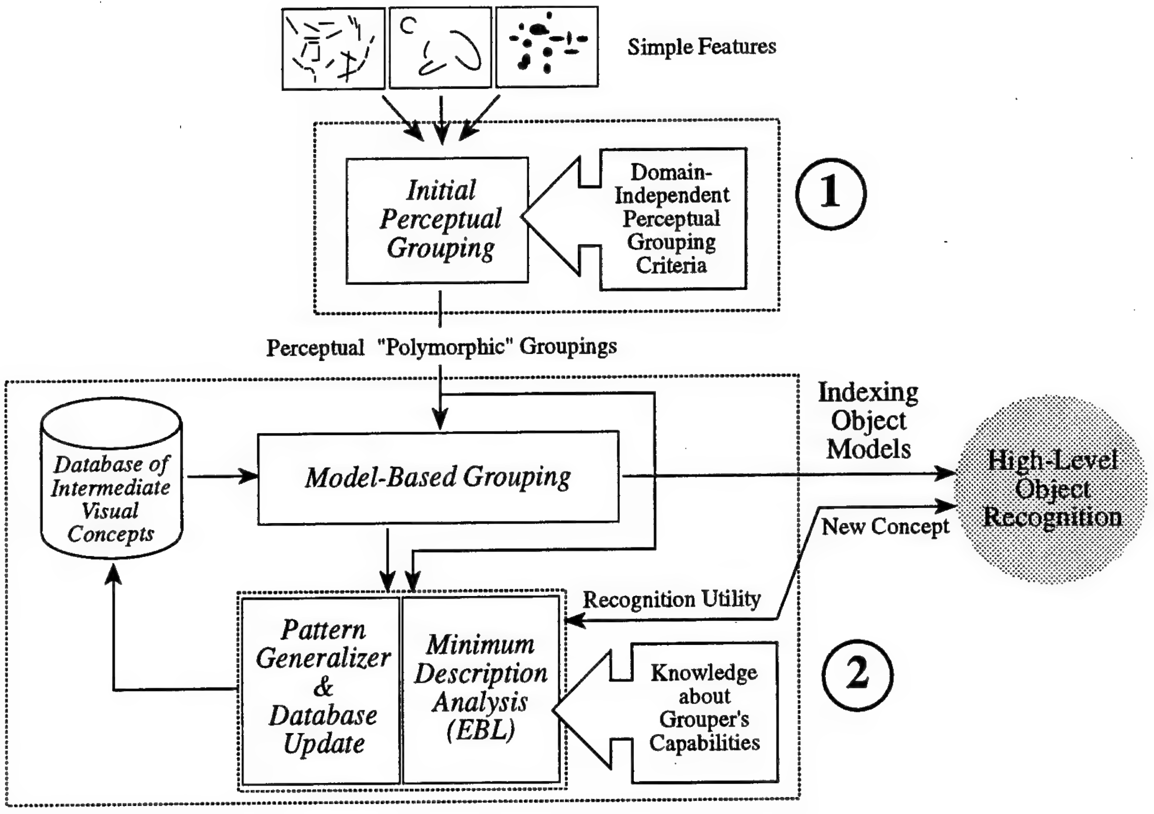

Simple Features
Initial Perceptual Grouping
Domain-Independent Perceptual Grouping Criteria
1
Perceptual "Polymorphic" Groupings
Database of Intermediate Visual Concepts
Model-Based Grouping
Indexing Object Models
High-Level Object Recognition
New Concept
Recognition Utility
Pattern Generalizer & Database Update
Minimum Description Analysis (EBL)
Knowledge about Grouper's Capabilities
2

Figure 9.2: Learning intermediate visual concepts using Explanation-Based Learning (EBL).

(routine) recognition, the visual concepts found at this stage are directly used for indexing into the object model base.

Learning of new visual concepts is based on the following criteria:

Perceptual saliency: A concept must be perceptually salient, i.e., receive a high score in the first (perceptual) grouping stage.

Operationality: A concept must be describable in terms of the operators that the model-based grouper is able to perform. For this purpose, knowledge about these operators is supplied in explicit form.

Simplicity: Concepts that permit a simple description (i.e., one with few grouping steps

/ transformations) are preferred. EBL is used to find the simplest description for a given feature configuration (*Minimum Description Analysis*).

Recognition utility: Only those concepts that are found to be useful in recognizing a particular object are eventually accepted. This is determined by considering the outcomes of the high-level recognition steps.

Visual concepts in the database are generalizations of the actually observed feature configurations, produced by analytic (EBL) learning (Pattern Generalizer). The representation of a concept in the database is an annotated symbolic description, which is generalized by parameterizing specific geometrical properties of the corresponding feature representation. The task of the Model-Based Grouper module is to instantiate the visual concepts, in the stream of perceptual groups, operating in a goal-directed fashion. The concepts (goals) are supplied to the grouper as decision structures that are updated dynamically when the contents of the database are changed. Interaction with high-level object recognition occurs in two forms. First, instantiated known groups can be directly used for indexing into the model base at the high level. (The association between intermediate concepts and object models is done at the high level.) Secondly, high-level recognition is invoked to determine the recognition utility of new concepts.

The use of a small set of fixed bottom-up composite structural concepts allows efficient detection in images. Similar arguments hold for top-down search for specific arrangements when the number of possible objects is small. The disadvantage of this approach is that a small but fixed set of intermediate structural concepts is generally not useful in different application domains. For using top-down, model-based composite structures, the number of models is restricted. In both cases, the manual specification of suitable intermediate structures is difficult.

The following specific tasks are involved:

9.3.1 Task 1 — Model-Based Interpretation of Perceptual Groups

We develop methods for collecting structural primitives of different types (e.g., lines, arcs, parametric curves, blobs) into polymorphic groups, using a set of perceptually significant spatial relationships. The relationships (e.g., proximity, collinearity, symmetry, relative size) being used depend upon the type of elements contained in each particular group. The purpose of this initial bottom-up grouping process is to supply an ordered set of composite structures that have a high probability of being semantically meaningful. The database of perceptual relationships used in this task is fixed, i.e., not subject to adaptation during runtime. However, this database must be designed to allow easy extension when new structural feature types are introduced. The main subtasks are to develop (a) the database of perceptual relationships, (b) evaluation function to measure the "saliency" of high-order polymorphic groups, and (c) efficient grouping algorithms that can handle polymorphic structures.

9.3.2 Task 2 — Composite Structure Model Acquisition and Refinement

We consider the actual semantic significance of perceptual groups with regard to the given application domain, in contrast to the previous task, where we employ only general perceptual cues. The module developed in this task uses the initial perceptual groups developed in Task B.1 for ultimately creating an index into the object model database. For this purpose, the module tries to form more complex groups from the incoming simple groups by using a database of semantically relevant structures. The database is created and maintained by a learning scheme based on Explanation-Based Learning (EBL). The major steps in this task are (a) the development of a suitable representation for high-order polymorphic feature groups which can also express their variability, (b) the adaptation of EBL for learning parameterized geometric concepts and its implementation in software, and (c) the development of efficient matching algorithms that can make use of the polymorphic nature of the feature groups.

9.3.3 Task 3 — Composite Structure Learning Subsystem

The goal of this task is the integration of all components needed for the adaptive intermediate-level learning scheme. Here we address in particular the interaction between the database of composite feature structures (Task 2) and the object models at the high level. The interaction with the high-level recognition module is needed to determine the utility of an observed feature structure for recognizing a particular object.

9.4 Learning at the Intermediate-Level Vision: Previous Work

Learning at the intermediate level has been applied mainly in the areas of texture recognition, algorithm parameter adjustment, motion perception, and specific vision tasks, such as road following. Currently, clustering methods are the most popular adaptation or learning paradigm at this level, followed by the use of neural networks and some applications of genetic algorithms. Structural learning methods, such as EBL or CBR are currently much less used at the intermediate level.

An example for inductive learning at the intermediate level is the approach to texture recognition described by Pachowicz [61]. He uses a scaling process to convert feature vectors of texture statistics into symbolic intervals and then applies an inductive learning program to find the most preferred symbolic expression according to a specified criterion. The method also employs a rule optimization technique after texture learning and prior to recognition to allow rule generalization. A performance improvement over the traditional nearest-neighbor clustering method is demonstrated.

Gillies [29] reports a learning system based on Genetic Algorithms for generating image domain feature detectors to find the location of objects in the image. A genetic search method is used to generate populations of feature detectors which are morphological operators. The functions performed by the layered system are tailored to the specific imagery on which the system is trained. The system is also shown to handle multi-class discrimination.

Another application of Genetic Algorithms at the intermediate level is the work done by Roth and Levine [76], which is a learning-based approach to extraction of geometric primitives (parametric curves) from images. In this approach, a geometric primitive is genetically represented by the minimal set of points instead of its parameters. Learning involves determining the minimal set of points for a given primitive type that optimally fits the data. Montana [56] reports an expert system for the interpretation of passive sonar images that employs a GA for determining detection thresholds.

There is a growing number of neural network applications at the intermediate vision level. An example is the work by Pomerleau [68] on network-based navigation of autonomous robots. Due to their inability to capture and generalize structural descriptions, NNs in general do not appear to be well suited for solving structural problems at the intermediate level. There are, however, certain functional mapping problems at the intermediate level that can be addressed successfully with NNs. For example, Aloimonos and Shulman [3] have suggested the use of NNs to learn the parameters involved in "Shape-from-X" problems.

Intermediate-level composite structures are commonly detected by either bottom-up grouping criteria (see above) or specified *a priori* as prototype patterns that are searched for in a goal-directed manner (e.g., [57]). The work reported by Segen [82] addresses some aspects of learning composite structural concepts from examples, however, no results have been shown on real images. Structural feature detection is usually based on a fixed set of visual primitives for which efficient detection algorithms are available. The incorporation of features of varying complexity has been addressed using only fixed, domain-independent grouping criteria. The problem of automatically forming intermediate-level perceptual shape concepts has found considerable attention in the psychological field recently.

9.5 Explanation-Based Learning

Explanation-based learning (EBL) [20] is an extension to an earlier concept called "explanation-based generalization" described by Mitchell et al. in [54]. Both paradigms are based on the same idea of using strong domain knowledge to "explain" why a given training example is a member of the concept being learned.

The domain knowledge (or domain *theory*) required in EBL consists of three main components:

1. A specification of the *types* and *properties* of the objects being dealt with.
2. A set of inference rules for inferring relations and properties from given relations and properties, and possible transformations between objects in the domain.
3. A library of problem-solving operators (schemata) that were either learned from earlier training examples or are hand coded.

The learning task in EBL can be stated as finding a generalized sequence of legal transformations (a schema) to derive the goal configuration from a given initial configuration. This is usually accomplished in a two-step process:

1. Construct an explanation that is causal with respect to the domain knowledge. This is similar to constructing a proof sequence for a theorem with respect to a set of axioms.
2. Generalize that explanation into a new schema by looking for the weakest preconditions under which the same explanation would apply.

The main limitation of EBL in its original form lies in the fact that the domain knowledge must be complete. If a given training example cannot be explained in terms of the existing domain knowledge, no generalization and thus no learning can take place. Another issue is the way the domain knowledge is specified and used. In "pure" EBL, the domain knowledge is expressed in the form of first-order logic predicates or Horn clauses, which provide no notion of proximity or similarity in a quantitative sense. However, many domains require handling of approximate, distorted, or noisy descriptions, and are thus not well suited for EBL in its original form. As a consequence, there have been several suggestions for extending the capabilities of EBL, in particular for relaxing the problem of incomplete and and possibly incorrect domain knowledge by combining analytical (EBL) and inductive learning [83, 55, 63, 85].

A second shortcoming of EBL is its strong dependence of a "good" encoding of the domain theory rules, which makes it difficult to design a domain theory that produces correct specializations. One approach for solving this problem is to employ a weaker semantic bias when searching for a solution path, which, however, requires the use of multiple training examples (EBL can, in principle, produce generalizations from single training examples) [25].

9.6 EBL and Visual Concepts

In this section, we describe the principles of applying EBL in the context of structural feature analysis and visual concept acquisition. The first step is to define the basic elements of the EBL paradigm, i.e., objects, relations, inference rules, initial state, and goal state in terms of the structural feature domain.

9.6.1 Elements of the Learning Problem

The primitives involved in this learning approach are two-dimensional geometric primitives. The assumption is that we have suitable mechanism available for extracting these primitives from images. Primitive classes include zero-dimensional primitives (points), one-dimensional primitives (straight line segments, arcs), and fully two-dimensional primitives (closed curves, elliptical regions, parametric blobs, etc.), as indicated in Figure 9.2. We call these three primitive classes $\mathcal{P}_0$, $\mathcal{P}_1$, and $\mathcal{P}_2$, respectively.

The domain knowledge in this case consists of:

1. the properties of the individual primitives,
2. the spatial relations between primitives, and
3. a set of operators for combining (grouping) primitives into more complex arrangements.

The knowledge can be interpreted as a picture language (or algebra) for describing almost arbitrary configurations of picture primitives. In general, there is more than one possible description for a given arrangement of picture primitives. The *learning problem* consists of finding the simplest description (or a small *set* of simple descriptions) for a given picture configuration with respect to the current domain knowledge. The simplified descriptions found in the learning process become new intermediate-level visual concepts that are added to the current domain knowledge and can, in turn, become part of other object descriptions.

To evaluate the complexity of a particular description, each operator is associated with a cost term that represents the complexity of applying that operator or transformation. A similar approach is used in most approximate string matching techniques, where certain costs are associated with each character insertion, deletion, and replacement to compute a minimum "string edit" distance. The individual operator costs are assumed to be predefined and constant, at least originally. The questions of (a) how the operator costs should be related to the actual recognition mechanism and (b) if they can and should be learned pose interesting research topics.

9.7 Future Work

The work towards visual concept learning described in this chapter is still in an initial phase. Currently, our short-term goal in this problem area is to formalize the learning problem in precise terms and to specify suitable representations, learning algorithms, and performance measures. The plan is to adapt existing learning tools to this specific problem and to integrate these tools with other software components wherever possible. In addition, we are currently creating the necessary low-level operators for extracting structural features of various types that will allow to perform initial experiments on actual image data.

Bibliography

[1] D. Ackley. Stochastic iterated genetic hillclimbing. Technical Report CMU-CS-87-107, Carnegie Mellon University, Dept. of Computer Science, March 1987.

[2] J.A. Adam. Peacekeeping by technical means. *IEEE Spectrum*, 23(7):42–56, July 1986.

[3] J. Aloimonos and D. Shulman. Learning shape computations. In *Proc. DARPA Image Understanding Workshop*, pages 862–873, 1987.

[4] K.D. Ashley. Arguing by analogy in law: A case-based model. In D.H. Helman, editor, *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy.* Boston, MA: Kluwer, 1988.

[5] K.D. Ashley and E. Rissland. A case-based approach to modeling legal expertise. *IEEE Expert*, Summer 1988.

[6] A.G. Barto, R.S. Sutton, and C.J.C.H. Watkins. Learning and sequential decision making. Technical report, Univ. of Massachusetts, Amherst, MA, 1989.

[7] B. Bhanu. Automatic target recognition: State of the art survey. In *IEEE Trans. Aerospace and Electronic Systems*, volume 22 of *AES*, pages 364–379, 1986.

[8] B. Bhanu, X. Bao, and J. Peng. Reinforcement learning integrated image. In *Proceedings of DARPA Image Understanding Workshop*, New Orleans, LA, May 1997.

[9] B. Bhanu, R.N. Braithwaite, W. Burger, S. Rong, and X. Wu. Gabor waveletets for automatic target detection and recognition, quarterly report to ARPA. Technical report, College of Engineering, University of California, Riverside, CA, September, 1994.

[10] B. Bhanu and T. Jones. Image understanding research for automatic target recognition. In *DARPA Image Understanding Workshop*, pages 249–259, 1992.

[11] B. Bhanu and T. Jones. Image understanding research for automatic target recognition. *IEEE Trans. on Aerospace and Electronic Systems*, 8(10):15–23, Oct. 1993.

[12] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation.* Kluwer Academic Publishers, Boston, MA, Summer 1994.

[13] B. Bhanu, S. Lee, and S. Das. Adaptive image segmentation using genetic and hybrid search methods. *IEEE Trans. on Aerospace and Electronic Systems*, July 1995.

[14] B. Bhanu, S. Lee, and J. Ming. Adaptive image segmentation using a genetic algorithm. *IEEE Trans. on Systems, Man and Cybernetics*, July 1995.

[15] B. Bhanu and J. Ming. Recognition of occluded objects: A cluster-structure algorithm. *Pattern Recognition*, 20(2):199–211, 1987.

[16] E.L. Bienenstock, L.N. Cooper, and P.W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal Neuroscience*, 2:32–48, 1982.

[17] D. Chapman. Intermediate vision: Architecture implementation, and use. *Cognitive Science*, 16:491–537, 1992.

[18] R.T. Chin and C.R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, pages 67–108, March 1994.

[19] S. Das, B. Bhanu, and C.C. Ho. Generic object recognition using multiple. In *Image and Vision Computing 14*, pages 323–338, 1996.

[20] G.F. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[21] K. DeJong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.

[22] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, pages 321–370, December 1994.

[23] B.A. Draper, C.E. Brodley, and P.E. Utgoff. Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):888–893, 1994.

[24] M.A. Fischler. On the representation of natural scenes. In A.R. Hanson and E.M. Riseman, editors, *Computer Vision Systems*. Academic, New York, 1978.

[25] N.S. Flann and T.G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4:187–266, 1989.

[26] K.S. Fu and J.K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[27] K. Fukushima, S. Miyake, and T. Ito. Neocognition: A neural network model for a mechanism of visual patte rn recognition. In *IEEE Trans. on Systems, Man and Cybernetics*, pages 826–834, September and October 1983.

[28] D. Genter. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):411–436, 1983.

[29] A.M. Gillies. *Machine Learning Procedures for Generating Image Domain Feature Detectors.* PhD thesis, University of Michigan, Ann Arbor, MI, April 1985.

[30] D. Goldberg. *Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning.* PhD thesis, Dept. of Civil Engineering, University of Michigan, Ann Arbor, MI, 1983.

[31] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[32] D.E. Goldberg. Dynamic system control using rule learning and genetic algorithms. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 588–592, 1985.

[33] D.E. Goldberg and J.H. Holland. Special issue on genetic algorithms. *Machine Learning*, 2/3, 1988.

[34] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. Systems, Man, and Cybernetics*, 16(1):122–128, January 1986.

[35] J.J. Grefenstette, R. Gopal, B.J. Rosmaita, and D. VanGucht. Genetic algorithm for the traveling salesman problem. In *Proc. of Intl. Conf. Genetic Algorithms and Their Applications*, pages 160–168, July 1987.

[36] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692, 1990.

[37] V. Gullapalli. Associate reinforcement learning of real-valued functions. Technical report, Department of Computer and Information Science, University of Massachusetts, Amherst, May, 1993.

[38] R.M. Haralick. Performance characterization protocol in computer vision. In *Proc. Performance Versus Methodology in Computer Vision*, pages 26–32, Seattle, WA, June 1994.

[39] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:100–132, 1985.

[40] R.M. Haralick and L.G. Shapiro. Segmentation of images having unimodal distributions. In *Computer Vision, Graphics and Image Processing*, pages 100–132, 1985.

[41] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[42] J. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume II, pages 593–623. Morgan Kaufman, Los Altos, CA, 1986.

[43] N. Intrator and L.N. Cooper. Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5:3–17, 1992.

[44] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[45] J.L. Kolodner, R.L. Simpson, and K. Sycara. A process model of case-based reasoning in problem solving. In *Proc. 9th Intl. Joint Conf. Artificial Intelligence.* Los Angeles, CA, 1985.

[46] K.I. Laws. The phoenix image segmentation system: Description and evaluation. Technical Report 289, SRI International, Dec. 1982.

[47] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[48] R. Linsker. Self-organisation in a perceptual network. *Computer*, 21(3):105–117, 1988.

[49] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[50] R. Maclin and J.W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, pages 251–281, 1996.

[51] V.R. Mandava, J.M. Fitzpatrick, and D.R. Pickens III. Adaptive search space scaling in digital image registration. *IEEE Trans. on Medical Imaging*, 8(3):251–262, 1989.

[52] J.L. Marroquin and F. Girosi. Some extensions of the k-means algorithm for image segmentation and pattern classification. *A.I. Memo No. 1390, MIT AI Lab*, 1993.

[53] D.L. Milgram. Region extraction using convergent series. *Computer Graphics and Image Processing*, pages 1–12, 1979.

[54] T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[55] T.M. Mitchell and A.B. Thrun. Explanation-based learning: A comparison of symbolic and neural network approaches. In *Machine Learning: Proc. of the Tenth International Conference*, pages 197–204, Amherst, MA, June 1993. Morgan Kaufmann.

[56] D.J. Montana. Empirical learning using rule threshold optimization for detection of events in synthetic images. *Machine Learning*, 5:427–450, 1990.

[57] T.N. Mudge, J.L. Turney, and R.A. Volz. Automatic generation of salient features for the recognition of partially occluded parts. *Robotica*, 5:117–127, 1987.

[58] K.S. Narendra and M.A.L. Thathatchar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[59] R. Ohlander, K. Price, and D.R. Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[60] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[61] P.W. Pachowicz. Integrating low-level features computation with inductive learning techniques for texture recognition. *Int'l. Journal Pattern Recognition and Artificial Intelligence*, 4(2):147–165, 1990.

[62] T. Pavlidis. Springer-verlag, berlin-heidelberg-new york. *Structural Pattern Recognition*, 1977.

[63] M.J. Pazzani. Detecting and correcting errors of omission after explanation-based learning. In *Proc. IJCAI-89*, pages 713–718, 1989.

[64] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. In *Proc. DARPA Image Understanding Workshop*, pages 777–780, Monterey, CA, November 14-16 1994.

[65] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 538–543, San Francisco, CA, June 1996.

[66] J. Peng and B. Bhanu. Delayed reinforcement learning for closed-loop object recognition. In *Proc. of the 13th Int'l Conference on Pattern Recognition*, pages 310–314, Vienna, Austria, August 1996.

[67] J. Peng and R.J. Williams. Incremental multi-step q-learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 226–232, New Brunswick, NJ, 1994.

[68] D.A. Pomerleau. Neural network based autonomous navigation. In C. Thorpe, editor, *Vision and Navigation: The Carnegie Mellon Navlab*, pages 83–93. Boston, MA: Kluwer, 1990.

[69] V. Ramesh. *Performance characterization of image understanding algorithms.* PhD thesis, University of Washington, Ann Arbor, MI, 1995.

[70] J. Rasure and D. Argiro. *Khoros User's Manual.* University of New Mexico, 1991.

[71] B. Ravichandran. 2D and 3D model-base matching using a minimun representation criterion and hybrid genetic algorithm. Technical Report 105, Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute, Troy, NY, 1993.

[72] G. Reynolds and J.R. Beveridge. Searching for geometric structure in images of natural scenes. In *DARPA IU Workshop*, pages 257–271, Los Angeles, CA, 1987.

[73] C. Riesbeck and R. Schank. *Inside Case-Based Reasoning.* Hillsdale, NJ: Erlbaum, 1989.

[74] S. Rong and B. Bhanu. Characterizing natural backgrounds for target detection. In *Proc. ARPA Image Understanding Workshop*, Monterey, CA, November 14-16 1994.

[75] S. Rong and B. Bhanu. Enhancing a self-organizing map through near-miss injection. In *Proc. of the 1995 World Congress On Neural Networks*, pages I552–I556, Washington, D.C., 1995.

[76] G. Roth and M.D. Levine. A genetic algorithm for primitive extraction. In *Proc. Int'l. Conf. Genetic Algorithms*, pages 487–494, San Diego, CA, July 1991.

[77] G. Roth and M.D. Levine. Geometric primitive extraction using a genetic algorithm. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 640–643, Champaign, IL, June 1992.

[78] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Network*, 2:459–473, 1989.

[79] E. Saund. Symbolic construction of a 2-d scale-space image. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):355–395, 1990.

[80] R. Schank. *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge, MA: Cambridge University Press, 1982.

[81] R. Schank and R. Abelson. *Scripts, Plans, Goals, and Understanding.* Hillsdale, NJ: Lawrence Erlbaum, 1977.

[82] J. Segen. Learning structural descriptions of shape. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 96–99, 1985.

[83] A.M. Segre, editor. *Workshop on Combining Empirical and Explanation-Based Learning*, Proc. 6th International Workshop on Machine Learning, pp. 1–93, San Mateo, CA, 1989. Morgan Kaufmann.

[84] S. Shafer and T. Kanade. Recursive region segmentation by analysis of histograms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1166–1171, 1982.

[85] J.W. Shavlik and G.G. Towell. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3):231–253, 1989.

[86] J.W. Sherman, D.N. Spector, C.W. "Ron" Swonger, L.G. Clark, E.G. Zelnio, M.J. Lahart, and T.L. Jones. Automatic target recognition systems. In L. Shumaker, editor, *The Infrared and Electro-optical Systems Handbook*, pages 343–402. SPIE Optical Engineering Press, 1993.

[87] P. Suetens, P. Fua, and A.J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–59, 1992.

[88] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[89] S. Thrun and A. Schwartz. Segmentation of images having unimodal distributions. In *Proc. of Advances in Neural Information Processing Systems 7*, pages 385–392, Denver, CO, 1994.

[90] J.R. Ullmann. Edge replacement in the recognition of occluded objects. In *Pattern Recognition 26(12)*, pages 1771–1784, 1993.

[91] S. Wang and T. Binford. Local step edge estimation - a new algorithm, statistical model, and performance evaluation. In *Proc. of ARPA Image Understanding Workshop*, pages 1063–70, April 1993.

[92] C.J.C.H. Watkins. *Learning from delayed rewards.* PhD thesis, King's College, UK, 1989.

[93] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 3:229–256, 1992.

[94] R.J. Williams and J. Peng. Function optimatization using connectionist reinforcement learning. *Connection Science*, 3(3), 1991.

[95] P.H. Winston. *Artificial Intelligence.* Reading, MA: Addison-Wesley, 1984.

[96] Y.-J. Zheng. Feature extraction and image segmentation using self-organization networks. *Machine Vision and Applications*, 8(5):262–274, 1995.

[97] Y.-J. Zheng and B. Bhanu. Adaptive object detection based on modified hebbian learning. In *Proc. of 13th Int'l. Conference on Pattern Recognition*, volume 4, pages 164–168, August 1996.

[98] Y.-J. Zheng, W. Ritter, and R. Janssen. An adaptive system for traffic sign recognition. In *Proc. Intelligent Vehicles Symposium*, pages 165–170, Oct. 1994.